PingAccess

June 20, 2025

PINGACCESS
Version: 8.3;latest

Table of Contents

PingAccess

PingAccess protects web applications and APIs by applying security policies to client requests.

Release Notes

- Current
- Previous Releases

Get Started With PingAccess

- Introduction to PingAccess
- Installing and Uninstalling PingAccess
- Backing up and Restoring PingAccess
- Upgrading PingAccess

Use PingAccess

- PingAccess Use Cases
- Configuring and Customizing PingAccess
- PingAccess User Interface Reference Guide
- Reference Guides
- PingAccess Integrations
- PingAccess Monitoring Guide

Troubleshoot PingAccess

- Administrative SSO lockout
- Collecting support data
- Minimizing the PingAccess cookie size

Learn More

- PingAccess Community
- Support Portal
- PingAccess Customer Training (existing customers only)
- Partner Portal (partners)

Release Notes

New features and improvements in PingAccess. Updated June 20, 2025.

PingAccess 8.3 (June 2025)

Add custom claims to JWT identity mappings

New PA-15886

Map custom plain text values and return custom claims for anonymous users to give backend applications more context.

In the PingAccess admin console, a new section and checkbox are available in the JWT identity mapping advanced settings: **Custom JWT Claims** and **Create JWT for Anonymous Users**. Learn more in Creating JWT identity mappings.

Retry OIDC sign-on after an authorization code exchange failure

New PA-15920

Configure PingAccess web sessions and admin UI SSO authentication to allow OIDC sign-on reattempts if PingAccess can't exchange the authorization code for the ID and access tokens. When PingFederate authentication sessions are enabled, this can result in retries without requiring user interaction.

This provides a more seamless sign-on experience for autoscaling token provider environments and replaces vague error messaging about what to do next if an authorization code expires before the exchange.

> **Tip**
>
> The PingAccess log generates both an error message that the authorization code was invalid and a debug message that it's retrying the sign-on attempt.

You can configure how many times PingAccess should retry a sign-on attempt and how long it should wait between attempts. Learn more in the descriptions for the **Max Login Retries** and **Login Retry Delay (Sec.)** fields in advanced web session settings and configuring admin UI SSO authentication.

You can also configure the name of the temporary cookie created to track the current sign-on state. Learn more in the **Login State Cookie Name** in configuring web session management settings.

Enable PingAccess agents to authenticate with signed JWTs

New PA-15925, PA-15926, & PA-16050

Authenticate PingAccess agents to the engine nodes with a stronger authentication method.

> **Note**
>
> To use this feature, you must upgrade to PingAccess 8.2 or later and the PingAccess agent for Apache (RHEL or SLES) or NGINX 3.0 or later. Compatibility for the IIS and Apache (Windows) agents will be added in a future release.

After you configure a compatible PingAccess agent with the updated `agent.properties` file and select **Require Token Authentication** in the agent's configuration, the agent creates, signs, and sends a unique JWT for every authentication request.

> **Important**
>
> The JWT expires after 2 minutes, so you must ensure you synchronize the agent and the PingAccess server's clocks.

Learn more in the PingAccess 8.2 release notes. You can find setup instructions in Configuring PingAccess agents to use bearer token authentication and Agent SDK for C 3.0 (April 2025).

Use the PingAccess agent for NGINX with NGINX R33 or R34

New PA-16051 & PA-16052

Added support for NGINX R33 and R34. Learn more in NGINX agent system requirements.

Sign device profiles captured by PingAccess for the PingOne Protect integration

New PA-16054

Use the **Sign Device Profile** checkbox to sign and increase the security of device profiles captured by PingAccess and sent to PingOne Protect as part of the PingOne Protect integration. You can find configuration information in Risk policy field descriptions.

> **Note**
>
> Device profile signing can only be enabled when the **Device Profiling Method** is **Captured by PingAccess**.

Configure PingAccess to include the `typ` header in JWTs it creates

New PA-16059

You can now specify a value to use as the `typ` header for web sessions and JWT identity mappings with the **Type Header Value** field. PingAccess adds the configured header to the JWTs it generates unless you leave the field blank.

For example, when using a JWT access token, use the **Type Header Value** field to comply with IETF RFC 9068 - JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens. This enables resource servers to verify that the `typ` header value matches the recommended value of `at+jwt` (or `application/at+jwt`, if this value is required) for any configured JWTs.

You can find specific configuration guidance in the **Type Header Value** field description in Configuring advanced web session settings and Creating JWT identity mappings.

Write PingAccess logs in JSON format or to the console

New PA-16092

You can now write PingAccess logs in JSON format and can configure the logs to output to the console instead of a file. Learn more in Writing logs in JSON format.

Made PingAccess scripts POSIX compliant

Improved PA-15575

The `run.sh` and `obfuscate.sh` scripts are now POSIX compliant.

Reduce device profiling data collection for PingOne Protect integration

Improved PA-15921

PingAccess no longer collects user behavioral data by default when the **Device Profiling Method** is **Captured by PingAccess** because this subset of data wasn't rendered interactively in PingAccess. Streamlining data collection can reduce limitations caused by header size.

Upgraded BCFIPS library to version 2.0

Improved PA-15938

Upgraded to BCFIPS 2.0 for FIPS 140-3 compliance, resulting in the following changes:

- Two new properties are available in the `run.properties` file, `pa.trust.keystore.type` and `pa.trust.keystore.path`.

  Learn more about these properties in the **Configuration database and key store settings** section of the Configuration file reference.

- PingAccess no longer supports SHA-1 while running in FIPS mode.

  Learn more about PingAccess features that operate differently or are unavailable in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode. For example, PKCS#12 isn't a supported keystore type in FIPS mode.

Authenticate PingAccess agents without a shared secret

Improved PA-15967

PingAccess engine nodes can now authenticate bearer tokens sent by a PingAccess agent without requiring the shared secret to be sent as well.

By default, agents continue to send both the shared secret and the bearer token when the **Require Token Authentication** checkbox is selected. To prevent an agent from sending a shared secret, remove the `agent.engine.configuration.shared.secret` property from the `agent.properties` file you download.

Learn more about bearer token authentication in Configuring PingAccess agents to use bearer token authentication and Agent field descriptions.

Important

You can configure the agents with the new `agent.properties` file with no performance impact, but leave the **Require Token Authentication** checkbox cleared until both:

- Agent compatibility is added
- You've upgraded all agents to the supported version

Added configuration warning about conflicting cookie names

Improved PA-16081

Added a warning that configuring the **Cookie Name** and **Session State Cookie Name** web session management settings with the same value can lead to unexpected runtime behavior. For example, PingAccess might require you to reauthenticate for every request, or it might unset the cookie and remove the PingAccess session. Learn more in configuring web session management settings.

Fixed issues starting PingAccess in FIPS mode when using AWS CloudHSM

Fixed PA-15924

Fixed an issue that caused a `Null Pointer Exception` error when starting PingAccess in Federal Information Processing Standards (FIPS) mode if you had any AWS CloudHSM key pairs configured. This issue was also applicable if you tried to configure a new CloudHSM key pair while in FIPS mode.

- Learn more about FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.
- Learn more about AWS CloudHSM in Adding an AWS CloudHSM provider.

Fixed JSONPointer mapping to first header only when mapping statement responses

Fixed PA-15965

Fixed an issue that prevented referencing any statement after the first when mapping statement responses to headers for a PingAuthorize policy decision access control rule.

Note

Additionally, the rule now automatically adds the leading `/` to configured **Response JSON Pointer** values only if you haven't already included it.

Fixed inability to send a response body to PingAuthorize

Fixed PA-15966

Fixed inability to send an optional response body to PingAuthorize with the PingAuthorize policy decision access control rule.

Fixed upgrade blocker when upgrading from PingAccess 8.2.1 to 8.3

Fixed PA-16102

Fixed an issue that prevented PingOne Advanced Identity Cloud token provider configurations from migrating to PingAccess 8.3 when upgrading from PingAccess 8.2.1.

Fixed inability to change default CSP

Fixed PA-16035

Fixed an issue that prevented changing the default content security policy when using the **HTML OIDC Authentication Request** authentication challenge response generator.

Added the `pf.redirect.use.default.csp` property to the `run.properties` file. Learn more in the **Security headers properties** section of the Configuration file reference.

Fixed encryption validation for the `pa.keystore.pw` property

Fixed PA-16036

PingAccess now enforces encryption of the `pa.keystore.pw` property in the `run.properties` file. Learn more in Upgrade considerations.

Fixed inability to access Java trust store for CRL and OSCP trust anchor validation

Fixed PA-16037

Fixed an issue that prevented PingAccess from using certificate authorities (CAs) in the Java trust store for client certificate trust anchor validation when **CRL Checking** or **OSCP** is enabled and CAs haven't been configured within the trusted certificate group.

Fixed `pf.redirect.headers` CSP behavior

Fixed PA-16039

Fixed an issue that caused inconsistent behavior with default and custom content security policy (CSP) headers set by the `pf.redirect.headers` in the `run.properties` file.

Fixed infinite looping with PingOne Protect device profiling

Fixed PA-16080

Added a new advanced setting to the PingOne Protect policy configuration, **Max Device Profile Retries**. This configuration fixes an issue that causes PingOne Protect device profiling to get stuck in an infinite loop.

Learn more in Risk policy field descriptions.

Two admin API exclusive fields affected by admin console interaction

Issue PA-15863

Two fields that are exclusive to the PingAccess admin API are setting empty arrays as a default value whenever an admin saves a PingOne Advanced Identity Cloud or admin token provider configuration in the PingAccess admin console. This could lead to unexpected behavior at runtime.

Key pairs cause SSL exception when using Luna HSM Client 10.8

Issue PA-16103

Key pairs stored in a Safenet Luna HSM cause SSL exceptions if using Luna HSM Client 10.8.

A potential workaround for this issue is to disable TLS 1.3 and RSASSA-PSS in the `run.properties` file. You can find more information in the **TLS/SSL** section of the PingAccess Configuration file reference.

PingAccess can't shut down when using Luna HSM Client 10.8

Issue PA-16104

PingAccess fails to shut down when the Safenet Luna HSM `libCryptoki2.so` directory is in the `deploy` directory, which is a deployment requirement for Adding a Safenet Luna provider on a Linux system. This is an issue specific to Luna HSM Client 10.8.

PingAccess 8.2.1 (April 2025)

Configure PingAccess to retry failed target site connections immediately

New PA-15943

The minimum value for the **Failed Retry Timeout (S)** field in PingAccess availability profile configurations is now `0` instead of `1`. This lets you to remove the delay before PingAccess retries establishing a connection to a failed target site.

Learn more in Creating availability profiles.

BCFIPS library upgraded to version 2.0

Improved PA-15938

Upgraded to BCFIPS 2.0 for FIPS 140-3 compliance, resulting in the following changes:

- Two new properties are available in the `run.properties` file, `pa.trust.keystore.type` and `pa.trust.keystore.path`.

  Learn more about these properties in the **Configuration database and key store settings** section of the Configuration file reference.

- PingAccess no longer supports SHA-1 while running in FIPS mode.

Learn more about PingAccess features that operate differently or are unavailable in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode. For example, PKCS#12 isn't a supported keystore type in FIPS mode.

Authenticate PingAccess agents with bearer tokens only

Improved PA-15967

PingAccess engine nodes can now authenticate bearer tokens sent by a PingAccess agent without requiring the shared secret to be sent as well.

By default, agents continue to send both the shared secret and the bearer token when the **Require Token Authentication** checkbox is selected. To prevent an agent from sending a shared secret, remove the `agent.engine.configuration.shared.secret` property from the `agent.properties` file you download.

Learn more about bearer token authentication in Configuring PingAccess agents to use bearer token authentication and Agent field descriptions.

Important

The PingAccess agents haven't been updated to support bearer token authentication yet. You can configure the agents with the new `agent.properties` file with no performance impact, but leave the **Require Token Authentication** checkbox cleared until both:

- Agent compatibility is added
- You've upgraded all agents to the supported version

Fixed an issue with post-authentication method type expectations

Fixed PA-15762

We've fixed an issue that caused requests to fail because of resource method enforcement.

PingAccess disables request preservation for the templated, redirect, and PF Authentication API challenge response generators, expecting the frontend SPA to maintain any data that requires preservation. As a result, PingAccess was expecting a `GET` request after authentication instead of a `POST` request because PingAccess only maintains post-authentication requests as a `POST` if request preservation is enabled.

Fixed inability to change a default CSP

Fixed PA-16035

We've fixed an issue that prevented changing the default content security policy when using the **HTML OIDC Authentication Request** authentication challenge response generator.

We've also added the `pf.redirect.use.default.csp` property to the `run.properties` file. Learn more in the **Security headers properties** section of the Configuration file reference.

Fixed issues starting PingAccess in FIPS mode when using AWS CloudHSM

Fixed PA-15924

We've fixed an issue that caused a `Null Pointer Exception` error when starting PingAccess in Federal Information Processing Standards (FIPS) mode if you had any AWS CloudHSM key pairs configured. This issue was also applicable if you tried to configure a new CloudHSM key pair while in FIPS mode.

- Learn more about FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.
- Learn more about AWS CloudHSM in Adding an AWS CloudHSM provider.

Fixed issues starting PingAccess in FIPS mode when using Oracle JDK 17 or 21

Fixed PA-15935

We've fixed an issue that caused PingAccess to fail to start in FIPS mode when using Oracle JDK 17 or 21.

PingAccess 8.2 (December 2024)

PingAccess for Azure AD program ends in December 2025

Info PA-15870

> **Important**
>
> The PingAccess for Azure AD program ends on December 31, 2025. To continue using PingAccess, you must upgrade to a commercial PingAccess license. Learn more in:
>
> - PingAccess for Azure AD Overview for an overview of license differences
> - Manage license keys
> - View or upload a new license

Create custom log level categories

New PA-15743

Add a custom log level category and manage its verbosity in the admin console. Learn more in Creating custom log level categories.

Added support for Java 21

New PA-15765

- Added support for Java 21. Learn more in System requirements.

- Updated Managing Federal Information Processing Standards (FIPS) mode to include more information about default TLS cipher suites and running PingAccess as a Windows service.

Configure PingOne Advanced Identity Cloud or PingAM as a token provider

New PA-15768

Configure PingOne Advanced Identity Cloud or PingAM as a token provider and OAuth authorization server in PingAccess. Learn more in Configuring PingOne Advanced Identity Cloud or PingAM as the token provider.

Configure an expected response header for CORS preflight requests

New PA-15766

Google Chrome cross-origin resource sharing (CORS) preflight requests will soon include a new request header, `Access-Control-Request-Private-Network: true`. If a preflight request that contains this header doesn't receive a `Access-Control-Allow-Private-Network: true` header in response, access requests will be denied.

To respond to CORS preflight requests with the expected response header, select the new checkbox in the PingAccess cross-origin request rule: **Allow Private Access Network**.

Configure `SameSite` settings on PingAccess nonce cookies

New PA-15803

Use the **Nonce SameSite Cookie** list to select a level of restriction for when nonce cookies can be sent in a cross-site request. Learn more in Configuring web session management settings.

Configure a PingAuthorize policy decision access control rule for fine-grained access control

New PA-15770

Added a new rule that makes use of the Policy Decision Endpoint in PingAuthorize. This enables more control over fine-grain authorization decisions sent to PingAuthorize than the PingAuthorize access control rule.

Learn more in Adding PingAuthorize policy decision access control rules.

> **Note**
>
> The PingAuthorize policy decision access control rule isn't compatible with PingOne Authorize.

Configure multiple JWKS endpoints for access token validation

New PA-15871

Added a new access token validator type, **Multiple JSON Web Key Set (JWKS) Endpoint**. This access token validator enables you to validate incoming access tokens from multiple authorization servers.

Learn more in Adding access token validators.

Configure PingAccess to allow agents to authenticate with a bearer token

New PA-15872

Authenticate PingAccess agents to the engine nodes with a stronger authentication method. Learn more in Configuring PingAccess agents to use bearer token authentication.

Added a new checkbox to the agent configuration page in the PingAccess administrative console: **Require Token Authentication**. This checkbox configures the PingAccess engine nodes for bearer token authentication. Learn more in Agent field descriptions.

> **Important**
>
> Only the PingAccess agent for Apache (RHEL), the PingAccess agent for Apache (SLES), and the PingAccess agent for NGINX have been updated to support bearer token authentication currently. You can configure the Apache (Windows) and IIS agents with the new `agent.properties` file with no performance impact, but leave the **Require Token Authentication** checkbox cleared until both:
>
> - Agent compatibility is added
> - You've upgraded all agents to the supported version

Added support for Amazon Linux 2023

New PA-15783

Added support for Amazon Linux 2023. Learn more in System requirements.

Configure PingAuthorize access control and response filtering rules with PingOne Authorize

Improved PA-15790

The PingAuthorize access control and response filtering rules are now compatible with PingOne Authorize, with the following limitations:

- PingAuthorize access control rule: Make sure that the **Include Identity Attributes** checkbox is selected in step 7 of Adding PingAuthorize access control rules.
- PingAuthorize response filtering rule: Detailed request context isn't available during response processing, so response filtering can't be performed with the `PingOne.API Access Management.Identity.Access Token` attribute.

Fixed agent page behavior after downloading `agent.properties` in Firefox

Fixed PA-13704

Fixed an issue that caused the agent configuration page in the PingAccess administrative console to stop responding after a user downloaded the `agent.properties` file in Mozilla Firefox.

Fixed default value rendering

Fixed PA-15763

Fixed an issue that caused some authentication challenge policy (ACP) configuration fields to render their default value only after they were saved.

Fixed OIDC login failure when port 443 is used in the `id_token` issuer

Fixed PA-15772

Fixed an issue that caused `id_token` validation to fail because PingAccess didn't accept the well-known HTTPS port 443 in `id_token` issuers and wouldn't register the issuer as a match.

Fixed an issue with bearer token case-sensitivity

Fixed PA-15890

Fixed an issue that caused false `401` errors because PingAccess was processing bearer tokens case-sensitively. PingAccess has been updated to meet RFC 9110.

Fixed shared secret timestamps in agent summaries

Fixed PA-15896

Fixed an issue that caused the PingAccess administrative console to fail to display agent shared secret timestamps in the agent configuration summary.

Cannot assign rule sets containing a singular CORS rule

Issue PA-15785

Rule sets or rule set groups containing a singular CORS rule cannot be assigned to applications or resources. Attempts result in the following validation error:

```
Invalid rule assignment for Application '<app_name>': assigning multiple Cross-Origin Request Policies to a Resource or RuleSet is not allowed.
```

Saving overwrites the sslCiphers and sslProtocol fields in the administrative API

Issue PA-15863

Saving a configuration in the PingAccess administrative console overwrites the values of the API-only fields **sslCiphers** and **sslProtocols**.

This issue is only relevant for the following pages in the administrative console:

- **System > Token Provider** (with **PingOne Advanced Identity Cloud / PingAM** selected)
- **System > Admin Authentication > Admin Token Provider**

It affects the following administrative API endpoints:

- `/pingone/advancedIdentityCloud`
- `/auth/tokenProvider`

Cannot use FIPS mode with a AWS CloudHSM or Safenet Luna HSM

Issue | PA-15924 | PA-15928

Federal Information Processing Standards (FIPS) mode doesn't work with AWS CloudHSM or Safenet Luna HSM. Trying to configure a key pair or enter FIPS mode with a key pair already configured causes a `Null Pointer Exception` error.

ACME account creation fails while PingAccess is in FIPS mode

Issue | PA-15929

Federal Information Processing Standards (FIPS) mode cannot be used with ACME certificate management if you need to create an ACME account.

Cannot use FIPS mode with Oracle JDK 17 and 21

Issue | PA-15935

PingAccess fails to start in Federal Information Processing Standards (FIPS) mode when using Oracle JDK 17 and 21. Currently, FIPS mode can only be used with OpenJDK or Amazon Corretto.

PingAccess 8.1.3 (April 2025)

Fixed an issue with post-authentication method type expectations

Fixed | PA-15762

We've fixed an issue that caused requests to fail because of resource method enforcement.

PingAccess disables request preservation for the templated, redirect, and PF Authentication API challenge response generators, expecting the frontend SPA to maintain any data that requires preservation. As a result, PingAccess was expecting a `GET` request after authentication instead of a `POST` request, because PingAccess only maintains post-authentication requests as a `POST` if request preservation is enabled.

Fixed inability to change a default CSP

Fixed | PA-16035

We've fixed an issue that prevented changing the default content security policy when using the **HTML OIDC Authentication Request** authentication challenge response generator.

We've also added the `pf.redirect.use.default.csp` property to the `run.properties` file. Learn more in the **Security headers properties** section of the Configuration file reference.

PingAccess 8.1.2 (October 2024)

Create custom log level categories

New PA-15743

Add a custom log level category and manage its verbosity in the admin console. Learn more in Creating custom log level categories.

Fixed incorrect file path when running PingAccess as a Windows service

Fixed PA-15805

Fixed an issue with the `log4j-categories.xml` file path that caused logging to break when running PingAccess as a Windows service.

Fixed an issue with establishing WebSocket connections when using IWA

Fixed PA-15801

Fixed a processing behavior conflict that prevented WebSocket connections from being established when using Integrated Windows Authentication (IWA).

PingAccess 8.1.1 (August 2024)

Fixed a security vulnerability with URL-encoded characters

Security PA-15776

Added the `pa.uri.canonicalize` parameter to the Configuration file reference to fix a security vulnerability. Learn more in an upcoming security advisory.

Set response headers for OAuth errors

Improved PA-15764

Added the `oauth.error.headers` and `oauth.error.header.Content-Security-Policy` parameters to the Configuration file reference.

Fixed default value rendering

Fixed PA-15763

Fixed an issue that caused some authentication challenge policy (ACP) configuration fields to render their default value only after they were saved.

PingAccess 8.1 (June 2024)

Improved request header security

Security PA-15610

Fixed an issue with connection request header handling. Learn more in SECADV045.

PingAccess 9.0 will remove Java 11 support in December 2025

Info

Ping Identity intends to remove Java 11 support from PingAccess in December 2025. For more information, including Java 17 support, see Installation requirements.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types. A new PAAP header, `vnd-pi-token-cache-oauth-ttl`, helps the agent distinguish between cookie and authorization header cache TTLs.

When the application type is **Web + API**, PingAccess only returns the TTL header corresponding with the token-type that it used to make the access decision, `vnd-pi-token-cache-ttl` or `vnd-pi-token-cache-oauth-ttl`. For more information, see PAAP agent response, and, after upcoming agent releases, the `agent.cache.defaultTokenType` property in one of the following agent configuration pages:

- RHEL agent configuration
- SLES agent configuration
- Windows agent configuration
- IIS agent configuration

Previously, cache definitions were only maintained for the cookie token-type. Accordingly, the ability to cache cookie and authorization header token-types simultaneously improves system performance because an agent doesn't have to call the PingAccess server every time it receives a request with an authorization header.

> **Note**
>
> Existing agent environments ignore the new `vnd-pi-token-cache-oauth-ttl` header and additional paths in the `vnd-pi-resource-cache` header.
> To see the performance boost, upgrade to PingAccess 8.1 and, after upcoming agent releases, upgrade to the latest version of the desired Apache or IIS agent. Otherwise, continue to use an earlier agent version.

CEF logging

New PA-15579

Enable PingAccess to write any of its five audit logs in Common Event Format (CEF). Learn more in Writing audit logs in Common Event Format.

Skip the request or response payload in a PingAuthorize call

New PA-15585

You can now control whether to include the request body in a call to PingAuthorize or a response body in the modified response. Excluding request and response bodies improves performance if the request or response body isn't required to make an access decision or to modify the response.

Learn more in Adding PingAuthorize access control rules and Adding PingAuthorize response filtering rules.

Map static header values to a PingAuthorize call

New PA-15586

Declare headers that should be added to the PingAuthorize request or to the modified response. PingAuthorize uses the additional headers to determine the policy set that's most relevant to the request or response context.

Learn more in Adding PingAuthorize access control rules and Adding PingAuthorize response filtering rules.

Use PingAuthorize access control rules on agent applications and resources

New PA-15587

You can now use PingAuthorize access control rules in PingAccess agent deployments. Learn more in Adding PingAuthorize access control rules.

Set PingAccess cookies with the `partitioned` attribute

New PA-15588 and PA-15690

Added the ability to set PingAccess cookies with the `Partitioned` attribute to align with the Cookies Having Independent Partitioned State (CHIPS) specification. This helps maintain support for applications that use an iframe or other third-party embedded context to interact with web resources that PingAccess protects as browsers phase out third-party cookies that don't support CHIPS.

CHIPS enables third-party sites to continue to set cross-site cookies as long as they have the `Partitioned` attribute. To limit cross-site tracking, partitioned cookies can only be read within the same context of the top-level site where they were initially set. Learn more about CHIPS in https://developer.mozilla.org/en-US/docs/Web/Privacy/Privacy_sandbox/Partitioned_cookies and https://developers.google.com/privacy-sandbox/3pcd/chips.

In the PingAccess administrative console or API, use the **Partitioned Cookie** setting to control whether to add the `Partitioned` attribute to all cookies that PingAccess sets for a specific web session or the admin web session.

In the `run.properties` file, use the `pa.default.cookie.attributes.partitioned.excludedUserAgentPatterns` property to exclude the `Partitioned` attribute only when using a browser that doesn't support it. Learn more in the **Engine properties** section of the Configuration file reference.

> **Note**
>
> The `pa.default.session.cookie.attributes.partitioned` property is also new in PingAccess 8.1, but this property is moreso intended for potential future backports. In PingAccess 8.1 and later, the **Partitioned Cookie** setting is the preferred way to set the `Partitioned` attribute. The **Partitioned Cookie** value overrides the value of the `pa.default.session.cookie.attributes.partitioned` property.

Encrypt PingAccess cookies using AES-GCM encryption algorithms

New PA-15605

Added support for Advanced Encryption Standard Galois/Counter Mode (AES-GCM) encryption algorithms. Three AES-GCM encryption algorithms are now available in the **Encryption Algorithm** list on the **Web Session Management** page:

1. AES 128 with GCM
2. AES 192 with GCM
3. AES 256 with GCM

Learn more about configuring encryption algorithms in Configuring web session management settings.

Configure the `SameSite=Strict` attribute on web session cookies

New PA-15706

Added a new level of restriction to the **SameSite Cookie** list in advanced web session settings, `SameSite=Strict`. The `SameSite=Strict` attribute provides the strongest level of cross-site request forgery (CSRF) protection, but should not be configured as the sole means of defense against CSRF.

Learn more in Configuring advanced web session settings and https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-rfc6265bis-14#section-8.8.

Temporarily promote a replica admin node to the primary admin node

New PA-15707

Added two new endpoints to the replica admin console server:

- **GET /adminConfig/replicaAdmin/status**
- **POST /adminConfig/replicaAdmin/promote**

Made the following endpoints available from the replica admin console server:

- **GET /adminConfig/replicaAdmins/**
- **GET /adminConfig/replicaAdmins/{id}**

In DevOps environments, you can now temporarily promote the replica admin node through the replica admin API if the primary node is unavailable, but you must complete the Manually promoting the replica administrative node procedure to make this change permanent. This change provides greater availability for the PingAccess admin console by decreasing the amount of time it takes to promote the replica admin node. Learn more in Using the admin API to temporarily promote the replica administrative node.

Opt out of automatic URL encoding

Improved PA-15697

By default, redirect rules and rejection handlers automatically URL encode the admin input redirect URL. This could cause unexpected behavior if an application targeted by a redirect requires the URL to follow a specific format.

You can now opt out of automatic URL encoding by deselecting the **Encode URL** check box on a specific application resource logout or redirect response generator, redirect rule, redirect authentication challenge response generator, or redirect rejection handler. Learn more in:

- Adding application resources
- Adding redirect rules
- Authentication challenge response generator descriptions
- Creating rejection handlers

Fixed `NullPointerException` with the rewrite content rule

Fixed PA-15612

Fixed an issue that caused a `NullPointerException` error when the rewrite content rule was used on a resource that returned an empty chunked response body.

Fixed an issue with the agent response returning a non-default token TTL for unprotected API resources

Fixed PA-15622

Fixed an issue that caused unprotected agent API resources to have unexpected OAuth TTL values.

Fixed an issue with accessing global unprotected resources

Fixed PA-15692

Fixed a regression issue that caused a `500` error response when accessing a global unprotected resource on a **Web + API** application.

Fixed issues with query parameter behavior due to automatic URL encoding

Fixed PA-15696

Fixed an issue with automatically URL encoding target redirect URLs that sometimes disrupted query parameter sort order or added a trailing `=` to the end of single value query parameters. This issue affected redirect rules, redirect rejection handlers, redirect virtual resources, logout virtual resources, and redirect authentication challenge policy response generators.

Fixed admin JWKS endpoint returning a `401` or `500` response instead of the OAuth key set

Fixed PA-15723

Fixed an issue that caused PingAccess to override existing handling for the `/pa/oauth/JWKS` endpoint for the admin listener with the engine self-registration handler, prompting requests made to the endpoint to result in `401` unauthorized responses or `500` internal server errors.

Fixed engine self-registration failure

Fixed PA-15725

Fixed an engine self-registration failure issue caused by inability to determine if the engine self-registration token should be a Bearer or DPoP-bound token.

PingAccess 8.0.6 (April 2025)

Fixed inability to change a default CSP

Fixed PA-16035

We've fixed an issue that prevented changing the default content security policy when using the **HTML OIDC Authentication Request** authentication challenge response generator.

We've also added the `pf.redirect.use.default.csp` property to the `run.properties` file. Learn more in the **Security headers properties** section of the Configuration file reference.

PingAccess 8.0.5 (October 2024)

Create custom log level categories

New PA-15743

Add a custom log level category and manage its verbosity in the admin console. Learn more in Creating custom log level categories.

Fixed incorrect file path when running PingAccess as a Windows service

Fixed PA-15805

Fixed an issue with the `log4j-categories.xml` file path that caused logging to break when running PingAccess as a Windows service.

Fixed an issue with establishing WebSocket connections when using IWA

Fixed PA-15801

Fixed a processing behavior conflict that prevented WebSocket connections from being established when using Integrated Windows Authentication (IWA).

PingAccess 8.0.4 (August 2024)

Fixed a security vulnerability with URL-encoded characters

Security PA-15776

Added the `pa.uri.canonicalize` parameter to the Configuration file reference to fix a security vulnerability. Learn more in an upcoming security advisory.

Opt out of automatic URL encoding

Improved PA-15697

By default, redirect rules and rejection handlers automatically URL encode the admin input redirect URL. This could cause unexpected behavior if an application targeted by a redirect requires the URL to follow a specific format.

You can now opt out of automatic URL encoding by deselecting the **Encode URL** check box on a specific application resource logout or redirect response generator, redirect rule, redirect authentication challenge response generator, or redirect rejection handler. Learn more in:

- Adding application resources
- Adding redirect rules

- Authentication challenge response generator descriptions
- Creating rejection handlers

Set response headers for OAuth errors

Improved PA-15764

Added the `oauth.error.headers` and `oauth.error.header.Content-Security-Policy` parameters to the Configuration file reference.

Fixed issues with query parameter behavior due to automatic URL encoding

Fixed PA-15696

Fixed an issue with automatically URL encoding target redirect URLs that sometimes disrupted query parameter sort order or added a trailing `=` to the end of single value query parameters. This issue affected redirect rules, redirect rejection handlers, redirect virtual resources, logout virtual resources, and redirect authentication challenge policy response generators.

Fixed admin JWKS endpoint returning a `401` or `500` response instead of the OAuth key set

Fixed PA-15723

Fixed an issue that caused PingAccess to override existing handling for the `/pa/oauth/JWKS` endpoint for the admin listener with the engine self-registration handler, prompting requests made to the endpoint to result in `401` unauthorized responses or `500` internal server errors.

Fixed engine self-registration failure

Fixed PA-15725

Fixed an engine self-registration failure issue caused by inability to determine if the engine self-registration token should be a Bearer or DPoP-bound token.

PingAccess 8.0.3 (May 2024)

Fixed potential infinite loop issue with PingAccess clusters

Fixed PA-15741

Fixed a potential infinite loop issue that could prevent an engine node or replica administrative node from applying configuration changes.

PingAccess 8.0.2 (April 2024)

Configure CEF logging

New PA-15703

Enable PingAccess to write any of its five audit logs in Common Event Format (CEF). Learn more in Writing audit logs in Common Event Format.

PingAccess 8.0.1 (March 2024)

Improved request header security

Security PA-15610

Fixed an issue with connection request header handling. Learn more in SECADV045.

Fixed `NullPointerException` with the rewrite content rule

Fixed PA-157612

Fixed an issue that caused a `NullPointerException` error when the rewrite content rule was used on a resource that returned an empty chunked response body.

Fixed an issue with the agent response returning a non-default token TTL for unprotected API resources

Fixed PA-15622

Fixed an issue that caused unprotected agent API resources to have unexpected OAuth TTL values.

Fixed an issue with accessing global unprotected resources

Fixed PA-15692

Fixed a regression issue that caused a `500` error response when accessing a global unprotected resource on a Web + API application.

PingAccess 8.0 (December 2023)

PingAccess 8.0 upgrade notice - removed H2 dependency

Info PA-15358

If you have PingAccess 6.2 or below, you cannot upgrade directly to PingAccess 8.0. You must upgrade to a version above 6.2 first, and then upgrade to 8.0.

This is because in PingAccess 8.0, an outdated H2 JAR file was removed, and PingAccess 6.2 and below use an H2 embedded database.

Implement device profiling for PingOne Protect

New PA-15374

If you're using the PingOne Protect integration, you can now enable device profiling to implement attribute-based access control (ABAC) and enforce a complete zero trust strategy with PingAccess and PingOne Protect. You can:

- Set stricter constraints around when to perform a new risk evaluation.
- Automatically perform a new device profile collection and risk evaluation when an end user's IP address changes.
- Include device-related predictor types in the PingOne risk policy that you use for risk evaluation, including user and event behavior analytics and bot detection risk predictors. This enables you to use the default PingOne risk policy without needing to make any modifications and to trigger enforcement strategies like step-up authentication if abnormal device settings are detected.

Learn more about enabling device profiling in PingAccess in Risk policy field descriptions. Learn more about PingOne predictor types in Risk policies.

Important

Device profile collection adds the device profile to the user's browser as cookies, which are sent to PingAccess during subsequent requests. These cookies are usually 8192 bytes in size. Before enabling device profiling, you should increase the `pa.default.maxHttpHeaderSize` property in the `<PA_HOME>/conf/run.properties` file to ensure a smooth transition.

Use and validate OAuth 2.0 DPoP-bound access tokens

New PA-15517

Added the ability to use OAuth 2.0 Demonstrating Proof of Possession (DPoP) capabilities in a resource server role. This enables you to meet potential FAPI 2.0 Advanced Profile authorization server requirements in the future and prevent fraudulent access token usage.

Important

You won't be able to use DPoP with PingAccess unless both the OAuth API client and the token provider support DPoP as well.
As a security best practice, keep the value of the **DPoP Proof Lifetime (SEC.)** field low and consistent with the DPoP implementation of your API client anywhere that you configure DPoP settings in PingAccess.

Enable DPoP-bound access tokens in your token provider or admin token provider configuration. You can also override the global DPoP settings in your API authentication settings, or at the application or resource level for an `API` or `Web + API` application. For more information, see:

- If you're using PingFederate as the token provider, see Configuring OAuth resource servers. You must use PingFederate 11.3 or later.
- If you're using PingOne as the token provider, see Configuring PingOne.
- If you're using a common token provider, see Configuring OAuth authorization servers.
- To configure DPoP in your admin token provider settings, see Configuring an admin token provider.
- To override the global DPoP settings for API authentication, see Configuring API authentication.
- To override the global DPoP settings at the application level, see Application field descriptions.
- To override the global DPoP settings at the resource level, see Adding application resources.

Configure Microsoft Azure AD as a common token provider when protecting an API application

New PA-15369

PingAccess has made common token provider configuration more flexible:

- When you're configuring the OAuth authorization server for a common token provider, the **Introspection Endpoint** field is now required only if you configure a remote access token validator on your PingAccess application.
- When you're configuring an application, before you can select a remote access token validator from the **Access Validation** list, you must configure an **Introspection Endpoint** on the **OAuth Authorization Server** tab.

This increased flexibility enables you to configure Azure AD as the common token provider for protected API applications.

> **Important**
>
> Because Azure AD doesn't have an `introspection` endpoint and doesn't include a client ID value in tokens that they create, you must use a key from the `JWKS` endpoint to validate tokens locally when you're protecting an API application. For more information, see Configuring Azure AD as the common token provider when PingAccess is protecting an API application.

Filter applications by SPA support status

New PA-15375

- Added the ability to filter your applications by their SPA support status. For more information, see Editing an application.
- Added the `SPA Support` property to the **Properties** tab on PingAccess applications. You can now check whether an application has SPA support enabled by expanding the application instead of having to expand and open it.

Configure static signing keys for Private Key JWT

New PA-15376

By default, private key JWT OIDC code flow uses dynamic keys managed automatically by PingAccess. You can now opt to use static keys instead if you want to control key rotation yourself. For more information, see Configuring static signing keys.

> **Note**
>
> PingAccess currently only supports JWT signing with static keys, but might support encryption in the future.

You can:

- Enable static keys and select a signing key from your list of configured key pairs on a new page in the administrative console, **Static OAuth/OIDC Keys**. Then select a **Signing Algorithm** on the associated web session.
- Complete your static key configuration at the token provider. Click **View Metadata** on the **Static OAuth/OIDC Keys** page to retrieve your JWKS information and submit this information to your token provider. Alternately, use the PingAccess admin API endpoint `GET /staticKeys/JWKS` to retrieve your JWKS information.

  > **Note**
  >
  > You must update your JWKS information at the token provider because static and dynamic keys use different JWKS endpoints in PingAccess.
  > For example, if you're using PingFederate as the token provider, you must update the **JWKS URL** field in your configured OAuth client.

Use Microsoft SQL Server 2022 for audit event storage

New PA-15510

Added support for Microsoft SQL Server 2022 to enable migration to SQL server versions included in Microsoft's mainstream support policy.

Use Server-Sent Events (SSE) to push information from protected resource servers to web clients

New PA-15511

Qualified support for server-sent events (SSE) in PingAccess. You can use WebSockets or SSE to facilitate communication between the requesting client and a protected site. SSE pushes real-time updates in one direction, from server to client, whereas WebSockets uses bidirectional communication.

> **Important**
>
> Follow the defined standards to signal PingAccess to establish an SSE connection to the backend server and listen for real-time events, such as:
>
> 1. Configure the backend server to send the header `Content-Type: text/event-stream` through PingAccess.
> 2. Or set the client's request header as `Accept: text/event-stream`.
>
> After it receives the appropriate header from either the backend server or the client request, PingAccess establishes an SSE connection to the backend server when it grants a user access to a protected resource. PingAccess processes and acts on each event it receives from the backend server, then pushes an update to the client through a separate SSE connection.

Configure Microsoft Azure AD as the token provider for administrative API OAuth

New PA-15518

Added support for OAuth tokens created by Microsoft Azure AD for administrative API OAuth. This improves account security for administrators with Microsoft Azure AD configured as the token provider and enables administrators to use their own accounts to make PingAccess API changes. Relaxed the following PingAccess requirements:

- If you're using either a common token provider or administrative token provider configuration, you can now use a local access token validator to bypass administrative API OAuth validation that checks whether the token provider supports the introspection endpoint. This is necessary because Microsoft Azure AD does not have an introspection endpoint.
- If the administrative token is validated by a local access validator, the administrative API OAuth no longer enforces whether an administrative token contains a `scope` claim with a configurable value, because Microsoft Azure AD uses a `scp` claim instead.

Map SAML tokens as HTTP request headers

New PA-15525

Added the ability to map the SAML token received from a SAML token mediator site authenticator to an HTTP request header that you specify instead of mapping the token as a request cookie. For more information, see the Logged In Header Name field.

Choose a case-matching strategy for Admin SSO and OAuth roles

New PA-15527

You can now choose a case-matching strategy for administrative single sign-on and OAuth roles, not just web session attribute rules. Selection options are:

- Case-sensitive
- Case-insensitive
- DN matching

For more information, see Configuring API authentication and Configuring admin UI SSO authentication.

Updated PingAccess documentation link to be version-specific

Improved PA-15378

Updated the **Help** icon link in the administrative console that takes you to the PingAccess documentation. In PingAccess 8.0 forward, this link will now take you to the version of the documentation that matches the version of PingAccess that you're using.

Improved error message for configuring a risk policy with invalid data

Improved PA-15399

Improved an error message caused by sending an admin API request to create or update a risk policy with invalid or missing data. The error message no longer returns a `NullPointerException` error.

Removed non-system fonts

Improved PA-15529

Removed old fonts from the PingAccess administrative console to improve user experience.

Fixed inaccurate OAuth endpoint description in the PingAccess administrative API documentation

Fixed PA-15241

Fixed inaccurate reference to the OAuth authorization server as the OpenID Connect provider in the `DELETE` method of the `oauth/authServer` endpoint.

Fixed `SniHandlerConfigBuilder` parameter keystore type declaration

Fixed PA-15270

Fixed an issue that caused the `SniHandlerConfigBuilder` to fail to declare a specific keystore type for the PingAccess `SslContext` server, which could result in PingAccess taking longer to start up if the target JVM's default keystore type was PKCS#12.

The `SniHandlerConfigBuilder` now specifically declares JKS as the keystore type to prevent unexpected performance losses.

Fixed UI rendering issue when optional field is missing from plugin

Fixed PA-15273

Fixed an issue that caused the PingAccess administrative console UI to fail to render if a newly added configuration field was missing from the plugin data that was saved previously.

For more information, see create your own plugins.

Fixed a race condition resulting in null values for replication data

Fixed PA-15380

Fixed an issue that caused unexpected behavior in PingAccess if you deleted an entity while a clustered console node was preparing a replication payload to share with other nodes in the cluster. Some examples of this unexpected behavior included:

- Hibernate throwing `EntityNotFoundExceptions` errors.
- PingAccess adding null objects to the replication payload. This behavior didn't always register as an error in the administrative console, but could still cause the replication data readers to throw exception errors.

Fixed UI rendering breakage when using Groovy script fields in composite plugin fields

Fixed PA-15381

Fixed an issue that caused the PingAccess administrative console UI to display a blank page if you attempted to configure a Groovy script field within a plugin entity in a composite field.

For more information, see create your own plugins.

Fixed form data registration of list fields in composite plugin fields

Fixed PA-15382

Fixed an issue that caused list fields embedded in composite plugin fields to register improperly in the form data for the PingAccess administrative console UI.

For more information, see create your own plugins.

Fixed object ID override of key pairs and certificates imported through the administrative API

Fixed PA-15386

Fixed an issue that caused PingAccess to replace object IDs defined on key pairs or certificates imported through the administrative API with an auto-generated object ID.

Additionally, the `POST /keyPairs/import` and `POST /certificates` API models have been updated to include more information on how to assign an ID for these object types.

Fixed log category preferences not sticking on restart

Fixed PA-15390

Fixed an issue that caused PingAccess to reset an environment's configured log setting categories on startup.

Fixed early expiration of cached PingOne Protect risk evaluation results

Fixed PA-15396

Fixed an issue with the PingOne Protect integration that caused PingAccess to calculate expiration values for cached risk evaluation results in milliseconds instead of seconds. This unexpected input value was disabling token caching after making a risk evaluation because PingAccess was receiving a false positive result that the risk evaluation cache data had expired.

Fixed Azure AD access token validation issue

Fixed PA-15496

Azure AD creates a `Application (Client) ID` value that exceeds 36 characters and automatically assigns that value as the `Audience` value in the access token. This prevented PingAccess from validating Azure AD access tokens because PingAccess previously accepted a maximum of 32 characters for an `Audience` value.

PingAccess can now accept a longer `Audience` value.

Fixed replication configuration identifiers updating before configuration changes were applied

Fixed PA-15506

Fixed an issue that caused PingAccess engine or replica admin nodes to update their replication configuration identifier before they had finished integrating changes into their runtime configuration. This would result in nodes using stale configuration information until a new configuration change event happened.

Fixed exclusion of admin API OAuth configuration from bulk export

Fixed PA-15537

Fixed an issue that caused admin API OAuth settings to be excluded from bulk export operations if you configure admin API OAuth with an access token validator but haven't set client credentials.

Fixed import failure caused by multiple trusted certificates in configuration

Fixed PA-15568

Fixed an issue that could cause PingAccess configuration imports to fail if you had multiple trusted certificates configured in your environment.

Spurious errors when installing PingAccess as a Windows service

Issue

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.

Zero downtime upgrade limitation

Issue PAPQ-1034

PingAccess 6.3 deployments that use the Sideband API feature cannot be upgraded using the zero downtime upgrade procedure. You must use a planned outage to upgrade such an environment.

TLS 1.3 limitation

Issue STAGING-8707

PingAccess may have difficulty maintaining TLS 1.3 connections when using JDK 11.0.0, 11.0.1, or 11.0.2 because of a defect in those versions. This might cause upgrades to fail on systems using these versions.

IPv6 limitation

Issue PA-1894

Incorrect handling for IPv6 literals in host header. Note that IPv6 is not currently supported.

Request preservation not supported with Safari private browsing

Issue PA-2896

Request Preservation is not supported with Safari Private Browsing.

Engine and Admin Replica connection issue

Issue PA-4888

Engines and admin replicas do not connect to admin console if a combination of IP addresses and DNS names are used.

Token processor issue

Issue PA-6262

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.

Unread message body handling

Issue PA-7068

In custom PingAccess plugins, using `com.pingidentity.pa.sdk.http.Message#setBody` or `com.pingidentity.pa.sdk.http.Message#setBodyContent` directly on an exchange's Response object to modify content from the backend can put PingAccess connections into indeterminate states. The workaround is to:

1. Either make a new instance or a copy of the Response object and modify body content in the copy.
2. Call `com.pingidentity.pa.sdk.http.Exchange#setResponse` with the new or copied request and response objects.

> **Note**
>
> `com.pingidentity.pa.sdk.http.Exchange#setResponse` discards the pending response body from the backend immediately. In a future release, a fix will be added to discard the response body only when PingAccess writes the response to the frontend.

Firefox limitation for time range rules

Issue PA-8651

Firefox does not correctly support the HTML5 time tag. When using the time range rule, enter time in 24-hour format.

Risk-based authorization rule issue during upgrade

Issue PA-10505

Upgrades will fail with a risk-based authorization rule if a third-party service is not used in the rule.

Virtual hosts with shared hostnames retention issue

Issue PA-11390

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.

Asynchronous front-channel logout issue

Issue PA-12647

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.

Invalid special characters permitted in identity mappings

Issue PA-13214

Invalid special characters (`(),/;<=>?@[\]\{}"`) can be added to the certificate to Header Mapping field in an identity mapping. Adding this identity mapping to an application will cause `400` errors when the application is accessed.

UI failure when assigning new key pair

Issue PA-13500

Assigning a new key pair to the Admin HTTPS listener if the browser does not trust the new key pair can prevent the UI from functioning. The workaround is to close the browser and re-open it so that all connections to the admin node use the new certificate.

Slow restarts in FIPS mode

Issue PA-14239

If PingAccess is repeatedly stopped and restarted in FIPS mode, subsequent restarts can take up to 5 minutes to complete. The workaround is to use a tool such as rng-tools to refresh `/dev/random` and make more entropy available faster. For example:

```
sudo yum install rng-tools
sudo rngd -b
```

CloudHSM limited in Java8u261

Issue PA-14414

CloudHSM functionality works in FIPS mode but not in regular mode for `Java8u261` and later. `RSASSA-PSS` signing algorithms fail with `Java8u261` or later, and HSM vendors and core Java use different naming conventions for the `RSASSA-PSS` algorithm. There is a documented workaround in Adding an AWS CloudHSM provider.

Kong API limitation

Issue PA-14466

Due to an outstanding defect in the Kong API Gateway, the `ping-auth` plugin currently does not support requests that utilize the `Transfer-Encoding` header. If PingAccess is used as the external authorization server, the rewrite content rule can prevent the page from displaying.

Certificate revocation list memory issue

Issue PA-14621

If a client certificate has a certificate revocation list (CRL) DistributionPoint that points to an extremely large CRL, PingAccess might suffer from high memory usage leading to Out of memory (OOM) exceptions.

Java 17 limitation

Issue PA-14863

BCFIPS and HSMs are not supported when using Java 17.

Spurious warning after upgrade or startup on Windows

Issue PA-14907

After starting PingAccess for the first time on a Windows system or upgrading PingAccess on a Windows system, a warning message is logged reporting that the `pa.jwk` file was not made non-executable. This message can be ignored.

Hibernate deadlock errors

Issue PA-14985

There are a few potential scenarios when the PingAccess data layer might encounter deadlocks. PingAccess should be able to recover from these deadlocks, so hibernate error logs can be ignored when followed by the log message `Recovered from database deadlock with transaction retry.`

Deadlock when importing applications with significant reuse

Issue PA-14978

A race condition caused by importing applications with significant reuse of virtual hosts or context roots can deadlock the Apache Derby DB.

PA-14974 added systematic deadlock handling to reattempt operations that lead to a deadlock condition in Apache Derby, but a specific fix for this deadlock scenario will be added in a future release to reduce wasted cycles and warning or error log messages.

Console Log Settings **page doesn't immediately reflect changes made in the API**

Issue PA-15351

If you have the administrative console and API open at the same time and you're on a console page that isn't **Log Settings**, the **Log Settings** page won't immediately populate any log changes that you make in the API.

To work around this issue, go to the **Log Settings** page. Perform a hard refresh, or go to another page and then return to **Log Settings**.

Mutual TLS with TLS 1.3 might not work with some target servers

Issue PA-15449

Mutual TLS with a backend site that requires post-handshake authentication is not supported when using TLS 1.3. Current workaround options are to remove the requirement for post-handshake authentication from the backend site or to disable TLS 1.3.

SNI isn't set up for virtual hosts only used in redirects

Issue PA-15559

Currently, SNI is only set up for virtual hosts that are actively configured in an application. This can prevent PingAccess from presenting an expected certificate for a given redirect host.

The workaround is to configure the source host in a redirect as the virtual host for a disabled PingAccess application.

Previous Releases

The release notes linked here show the changes in previous versions of PingAccess.

- PingAccess 8.2
- PingAccess 8.1
- PingAccess 8.0
- PingAccess 7.3
- PingAccess 7.2
- PingAccess 7.1
- PingAccess 7.0

- PingAccess 6.3
- PingAccess 6.2
- PingAccess 6.1
- PingAccess 6.0
- PingAccess 5.3
- PingAccess 5.2
- PingAccess 5.1
- PingAccess 5.0
- PingAccess 4.3
- PingAccess 4.2
- PingAccess 4.1
- PingAccess 4.0
- PingAccess 3.2
- PingAccess 3.1

PingAccess Use Cases

This section provides example uses cases for PingAccess deployments.

- Protecting a web application with PingAccess in a gateway deployment
- Protecting an API with PingAccess in a gateway deployment
- Protecting a web application with PingAccess in an agent deployment
- Protecting an API with PingAccess in an agent deployment
- Protecting an API with PingAccess in a sideband deployment

Protecting a web application with PingAccess in a gateway deployment

Protect a web application from unwanted access using PingAccess.

Prerequisites

Before configuring your PingAccess deployment to protect a web application:

- Install and run PingAccess. You can find the full procedure in Installing and Uninstalling PingAccess.
- Configure a token provider. The procedures vary depending on the token provider. Learn more in:
  - PingFederate.
  - PingOne.
  - Common token provider.

Steps

Complete the following steps to protect your web application:

1. Configure a virtual host.

   A virtual host represents the external face of the site you'll protect.

2. Configure a site.

   A site contains the internal details of the site you'll protect, including its actual location.

3. Configure a web session.

   A web session defines the details of how user credential information is retained. This lets the token provider authenticate the user when it's required for a protected application.

4. Configure a rule.

   Rules determine who can access what content and under which circumstances.

5. Configure an identity mapping.

   An identity mapping enables you to pass identity information to the protected application using headers.

6. Configure an application.

   An application joins the other pieces together, giving users access to the site according to the configured rules.

Configuring a virtual host

About this task

The virtual host is the external-facing portion of a web application. In a gateway deployment, the virtual host contains the host name and port that your users use to reach the protected web application.

For more information about this procedure, including optional steps that aren't included here, see Creating new virtual hosts.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click **+ Add Virtual Host**.
3. In the **Host** field, enter the name for the virtual host.

   This is the host name used by end users to reach the site, such as `myHost.com`.

   > **Note**
   >
   > You can use a wildcard (`*`) for part or all of the host name. For example, `*.example.com` matches all host names ending in .example.com, and `*` matches all host names.

4. In the **Port** field, enter the port number for the virtual host, such as `443`.
5. Click **Save**.

Configuring a site

About this task

A site is only used in a gateway deployment. It contains the target address for the protected web application and any other information necessary to access the application.

For more information about this procedure, including optional steps that aren't included here, see Adding sites.

Steps

1. Click **Applications**, then go to **Sites > Sites**.
2. Click **+ Add Site**.
3. In the **Site Name** field, enter a unique name of up to 64 characters, including special characters and spaces.

   This name is used internally.

4. In the **Targets** field, enter one or more targets.

   These targets are the actual locations of the site. The format for this is `hostname:port` or `IP address:port`, such as `www.example.com:80`.

5. Select the **Secure** check box if the site is expecting HTTPS connections.

> **Note**
>
> This decision depends on whether the target expects an HTTPS connections from the PingAccess system to the protected web application.
> If you select **Secure**, you must also select a **Trusted Certificate Group** from the list, or select **Trust Any** to trust any certificate presented by the listed targets. The trusted certificate group defines what certificates or issuing certificate authorities PingAccess will trust when acting as a client to the backend server.
> For information about importing a certificate and creating a trusted certificate group, see Importing certificates and Creating trusted certificate groups.

6. Click **Save**.

> **Note**
>
> If the target site can't be contacted, PingAccess saves the site and displays a warning indicating the reason the site couldn't be reached.

Configuring a web session

About this task

A web session specifies the details of how user information is stored.

For more information about this procedure, including optional steps that aren't included here, see Creating web sessions.

Steps

1. Click **Access**, then go to **Web Sessions > Web Sessions**.
2. Click **+ Add Web Session**.
3. In the **Name** field, enter a unique name for the web session, up to 64 characters, including special characters and spaces.
4. In the **Cookie Type** list, select **Encrypted JWT**.
5. In the **Audience** field, enter the audience that the PingAccess token is applicable to, represented as a short, unique identifier between 1 and 32 characters.

> **Note**
>
> PingAccess rejects requests that contain a PingAccess token with an audience that differs from what is configured in the web session associated with the target application.

6. In the **OpenID Connect Login Type** list, select **Code**.

> **Note**
>
> The **Code** login type is recommended for maximum security and standards interoperability, but other options are available. Learn more about the available profiles in step 6 of Creating web sessions.

7. In the **Client ID** field, enter the unique identifier (client ID) that was assigned when you created the OAuth relying party (RP) client within the token provider.

Learn more in Configuring OAuth clients in the PingFederate documentation.

8. In the **Client Credentials Type** list, select a client credentials type.

   Selecting a client credentials type is required when configuring the **Code** login type.

   Choose from:

   - **Secret**
   - **Mutual TLS**
   - **Private Key JWT**

   > **Note**
   > The OAuth client you use with PingAccess web sessions must have an OpenID Connect (OIDC) policy specified.
   > Learn more in Configuring OpenID Connect Policies.

9. Provide the information required for the selected credential type.

   Choose from:

   - **Secret** – Enter the **Client Secret** assigned when you created the OAuth relying party client in the token provider.
   - **Mutual TLS** – Select a configured **Key Pair** to use for Mutual TLS client authentication.
   - **Private Key JWT** – No additional information is required.

10. In the **Idle Timeout** field, specify the amount of time, in minutes, that the PingAccess token remains active when no activity is detected by the user.

    The default is `60` minutes.

    > **Note**
    > If there is an existing valid PingFederate session for the user, an idle timeout of the PingAccess session might result in its re-establishment without forcing the user to sign on again.

11. In the **Max Timeout** field, specify the amount of time, in minutes, that the PingAccess token remains active before expiring.

    The default is `240` minutes.

12. Click **Save**.

Configuring a rule

About this task

Rules are used to control the circumstances under which users can access the protected web server. Rules can grant or deny access based on criteria such as user parameters from the token provider, header values, network ranges, or web session attributes. You can configure any number of rules in your environment.

You can combine rules into rule sets, which combine multiple rules. You can configure rule sets to allow access to a resource if at least one rule's criteria is met or to only allow access if all rules have their criteria met. Access control rules are processed before processing rules. Each type of rule is otherwise processed in the order you specify when you create the rule set.

You can further combine rule sets into rule set groups, which combine multiple rule sets. As with rule sets, rule set groups can allow access if any one rule set's criteria are met or only if all rule sets' criteria are met. Rule sets are processed in the order you specify when you create the rule set group.

This example uses an HTTP request header rule to demonstrate how rules are created and used. Each environment has different requirements, and you can use any of the rules explained in the Rule management section according to your needs.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name.

   The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** list, select **HTTP Request Header**.
5. In the **Field** column, in the **Header** field, enter the HTTP header name you want to match in order to grant or not grant the client access.
6. In the **Value** field, enter the values 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 in the **Value** field, or add a wildcard after the host name (`host*` or `host:*`) to match what's in the HTTP request.
7. If you need additional header pairs, click **Add Row** to add an additional row, then repeat steps 5-6.
8. Click **Save**.

Configuring an identity mapping

About this task

A header identity mapping can expose one or more attribute values to the protected application in HTTP request headers.

For more information about this procedure, including optional steps that aren't included here, see Creating header identity mappings.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the mapping.

4. In the **Type** list, select **Header Identity Mapping**.
5. In the **Attribute to Header Mapping** section, in the **Attribute Name** field, enter the name of the attribute to retrieve from the user web session, such as `sub`.
6. In the **Header Name** field, enter the name of the header to contain the attribute value.

   > **Note**
   >
   > The HTTP request 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. In the **Certificate to Header Mapping** section, enter the header name included in 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.
8. Click **Save**.

Configuring an application

About this task

The application represents the protected web application as a whole. Including the virtual host and the site allows PingAccess to route requests directed at the front-end name to the correct back-end resource. By including a web session, you specify how user credential data is stored and for how long. After you create the application and its root resource, you can add one or more rules to control access to the protected web application.

Within the application, you can add one or more resources. Resources are specific components that require a different degree of security. You can apply different rules to a resource, letting you apply specific controls to portions of an application. This example procedure does not include resources. For information about adding resources to an application, see Configuring resource ordering in PingAccess.

For more information about this procedure, including optional steps that aren't included here, see Adding an application.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click **+ Add Application**.
3. In the **Name** field, enter a unique name for the application, up to 64 characters, including special characters and spaces.
4. In the **Context Root** field, enter the context root for the application.

   The context root represents the context at which the application is accessed at the site and 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.

5. In the **Virtual Host** list, select the virtual host you created.
6. In the **Application Type** section, click **Web**.
7. In the **Web Session** list, select the web session you created.
8. In the **Destination** section, click **Site**, then select the site you created.
9. Click **Save**.
10. Click **Applications**, then go to **Applications > Applications**.
11. Expand the new application in the list and click the **Pencil** icon.
12. Click the **Web Policy** tab.
13. Drag your rule from **Available Rules** onto the policy bar.
14. Click **Save**.

Protecting an API with PingAccess in a gateway deployment

You can protect an application programming interface (API) from unwanted access using PingAccess.

Prerequisites

Before configuring your PingAccess deployment to protect an API:

- PingAccess must be installed and running. You can find the full procedure in Installing and Uninstalling PingAccess.
- You must have a configured token provider. The procedures vary depending on the token provider. Learn more in:
  - PingFederate.
  - PingOne.
  - Common token provider.

Steps

Complete the following steps to protect your API:

1. Configure a virtual host.

   A virtual host represents the external face of the API you'll protect.

2. Configure a site.

   A site contains the internal details of the API you'll protect, including its actual location.

3. Configure a rule.

   Rules determine who can access what content and under which circumstances.

4. Configure an identity mapping.

   An identity mapping enables you to pass identity information to the protected application using headers.

5. Configure an application.

   An application joins the other pieces together, giving users access to the API according to the configured rules.

6. Configure a resource.

   A resource specifies an API endpoint and the methods that can be used to access it.

Configuring a virtual host

About this task

The virtual host is the external-facing portion of an API. In a proxy deployment, the virtual host contains the host name and port that your users use to reach the protected API.

For more information about this procedure, including optional steps that are not included here, see Creating new virtual hosts.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click **+ Add Virtual Host**.
3. In the **Host** field, enter the name for the virtual host.

   This is the host name used by end users to reach the site. For example, myHost.com. You can use a wildcard (`*`) for part or all of the host name. For example, `*.example.com` matches all host names ending in .example.com, and `*` matches all host names.

4. In the **Port** field, enter the port number for the virtual host. For example, `443` .
5. Click **Save**.

Configuring a site

About this task

A site is only used in a proxy deployment. It contains the target address for the protected API and any other information necessary to access the application.

For more information about this procedure, including optional steps that are not included here, see Adding sites.

Steps

1. Click **Applications**, then go to **Sites > Sites**.
2. Click **+ Add Site**.
3. In the **Site Name** field, enter a unique name of up to 64 characters, including special characters and spaces. This name is used internally.
4. In the **Targets** field, enter one or more targets.

These targets are the actual locations of the site. The format for this is `hostname:port` or `IP address:port`. For example, `www.example.com:80`.

5. Select the **Secure** check box if the site is expecting HTTPS connections.

> **Note**
>
> This decision depends on whether the target expects an HTTPS connections from the PingAccess system to the protected web application.
> If you select **Secure**, you must also select a **Trusted Certificate Group** from the list, or select **Trust Any** to trust any certificate presented by the listed targets. The trusted certificate group defines what certificates or issuing certificate authorities PingAccess will trust when acting as a client to the backend server. For information about importing a certificate and creating a trusted certificate group, see Importing certificates and Creating trusted certificate groups.

6. Click **Save**.

> **Note**
>
> If the target site cannot be contacted, PingAccess saves the site and a displays a warning indicating the reason the site could not be reached.

Configuring a rule

About this task

Rules are used to control the circumstances under which users can access the protected API. Rules can grant or deny access based on criteria such as user parameters from the token provider, header values, network ranges, or web session attributes. You can configure any number of rules in your environment.

You can combine rules into rule sets, which combine multiple rules. You can configure rule sets to allow access to a resource if at least one rule's criteria is met, or to only allow access if all rules have their criteria met. Access control rules are processed before processing rules. Each type of rule is otherwise processed in the order you specify when you create the rule set.

You can further combine rule sets into rule set groups, which combine multiple rule sets. As with rule sets, rule set groups can allow access if any one rule set's criteria are met, or only if all rule sets' criteria are met. Rule sets are processed in the order you specify when you create the rule set group.

This example uses an HTTP request header rule to demonstrate how rules are created and used. Each environment has different requirements, and you can use any of the rules explained in the Rule management section according to your needs.

Steps

1. Click **Access**, then go to **Rules** > **Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. From the **Type** menu, select **HTTP Request Header**.
5. In the **Field** column, in the **Header** field, enter the HTTP header name you want to match in order to grant or not grant the client access.

6. In the **Value** field, enter the values 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 in the **Value** field, or add a wildcard after the host name (`host*` or `host:*`) to match what is in the HTTP request.

7. If you need additional header pairs, click **Add Row** to add an additional row, then repeat steps 5-6.
8. Click **Save**.

Configuring an identity mapping

About this task

A header identity mapping can expose one or more attribute values to the protected API in HTTP request headers.

For more information about this procedure, including optional steps that are not included here, see Creating header identity mappings.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the mapping.
4. From the **Type** list, select **Header Identity Mapping**.
5. In the **Attribute to Header Mapping** section, in the **Attribute Name** field, enter the name of the attribute to retrieve from the user web session. For example, `sub` .
6. In the **Header Name** field, enter the name of the header to contain the attribute value.

> **Note**
>
> 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 by identifying the header as `User-Agent` , not `HTTP_USER_AGENT` .

7. In the **Certificate to Header Mapping** section, enter the header name included in 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.
8. Click **Save**.

Configuring an application

About this task

The application represents the protected API as a whole. By including the virtual host and the site, it allows PingAccess to route requests directed at the front-end name to the correct back-end resource. After you create the application and its root resource, you can add one or more rules to control access to the protected API.

Within the application, you can add one or more resources. Resources are specific components that require a different degree of security. You can apply different rules to a resource, letting you apply specific controls to portions of an application. This example procedure does not include resources. For information about adding resources to an application, see Configuring resource ordering in PingAccess.

For more information about this procedure, including optional steps that are not included here, see Adding an application.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click **+ Add Application**.
3. In the **Name** field, enter a unique name for the application, up to 64 characters, including special characters and spaces.
4. In the **Context Root** field, enter the context root for the API. This represents the context at which all of the API endpoints are accessed at the site.

   The context root 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.
5. From the **Virtual Host** list, select the virtual host you created.
6. In the **Application Type** section, select **API**.
7. In the **Destination** section, select **Site**, then select the site you created.
8. Click **Save**.
9. Click **Applications**, then go to **Applications > Applications**.
10. Expand the new application in the list and click the pencil icon .
11. Click the **Web Policy** tab.
12. Drag your rule from **Available Rules** onto the policy bar.
13. Click **Save**.

Configuring a resource

About this task

An application resource is a component within an application that requires a different level of security. By configuring resources for your API endpoints, you can add different rules for different endpoints and specify which methods are allowed.

For more information about this procedure, including optional steps that are not included here, see Adding application resources.

Steps

1. Click **Applications**, then go to **Applications** > **Applications**.
2. Click to expand the application you created in the previous procedure.
3. Click the **Pencil** icon.
4. Click the **Resources** tab.
5. Click **+ Add Resource**.
6. In the **Name** field, enter a unique name up to 64 characters, including special characters and spaces.
7. In the **Path Patterns** field, enter a list of Uniform Resource Locator (URL) path patterns, within the context root, that identify this resource.

   > **Note**
   >
   > The path pattern must start with a forward slash (/). It begins after the application context root and extends to the end of the URL.
   > When automatic path pattern evaluation ordering is in use (default), patterns can contain one or more wildcard characters (`*`). No use of wildcards is assumed. For example, there is a difference between `/app/` and `/app/*` . If a request matches more than one resource, the most specific match is used.

8. In the **Resource Authentication** section, select **Standard**, using the same authentication for the resource as for the root application.
9. In the **Methods** field, enter the methods supported by the resource.

   Leave the asterisk default if the resource supports all HTTP methods, including custom methods.

   > **Tip**
   >
   > Defining methods for a resource allows more fine-grained access control policies on API resources. If you have a server optimized for writing data (POST, PUT) and a server optimized for reading data (GET), you might want to segment traffic based on the operation being performed.

10. To log information about the transaction to the audit store, select the **Audit** check box.
11. To enable the resource, select the **Enabled** check box.
12. Click **Save**.

Protecting a web application with PingAccess in an agent deployment

You can protect a web application from unwanted access using PingAccess.

Prerequisites

Before configuring your PingAccess deployment to protect a web application:

- PingAccess must be installed and running. You can find the full procedure in Installing and Uninstalling PingAccess.
- You must have a configured token provider. The procedures vary depending on the token provider. Learn more in:
  - PingFederate.
  - PingOne.
  - Common token provider.
- You must have installed an agent on the web server or servers that host the site you want to protect. Learn more in:
  - PingAccess Agent for Apache (RHEL)
  - PingAccess Agent for Apache (SLES)
  - PingAccess Agent for Apache (Windows)
  - PingAccess Agent for IIS
  - PingAccess Agent for NGINX

Steps

Complete the following steps to protect your web application:

1. Configure a virtual host.

   A virtual host represents the site you'll protect and contains information about its location.

2. Configure a web session.

   A web session defines the details of how user credential information is retained. This lets the token provider authenticate the user when it's required for a protected application.

3. Configure a rule.

   Rules determine who can access what content and under which circumstances.

4. Configure an identity mapping.

   An identity mapping enables you to pass identity information to the protected application using headers.

5. Configure an application.

   An application joins the other pieces together, giving users access to the application according to the configured rules.

Configuring a virtual host

About this task

The virtual host is the external-facing portion of a web application. In an agent deployment, the virtual host contains the actual host name and port for the protected web application.

For more information about this procedure, including optional steps that are not included here, see Creating new virtual hosts.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click **+ Add Virtual Host**.
3. In the **Host** field, enter the name for the virtual host.

   This is the host name of the protected web application. For example, myHost.com. You can use a wildcard (`*`) for part or all of the host name. For example, `*.example.com` matches all host names ending in .example.com, and `*` matches all host names.
4. In the **Port** field, enter the port number for the virtual host. For example, `443` .
5. Click **Save**.

Configuring a web session

About this task

A web session specifies the details of how user information is stored.

For more information about this procedure, including optional steps that are not included here, see Creating web sessions.

Steps

1. Click **Access**, then go to **Web Sessions > Web Sessions**.
2. Click **+ Add Web Session**.
3. In the **Name** field, enter a unique name for the web session, up to 64 characters, including special characters and spaces.
4. From the **Cookie Type** list, select **Encrypted JWT**.
5. In the **Audience** field, enter the audience that the PA token is applicable to, represented as a short, unique identifier between one and 32 characters.

   > **Note**
   >
   > PingAccess rejects requests that contain a PA token with an audience that differs from what is configured in the web session associated with the target application.
6. From the **OpenID Connect Login Type** list, select **Code**.

   > **Note**
   >
   > The **Code** login type is recommended for maximum security and standards interoperability, but other options are available. Learn more about the available profiles in step 6 of Creating web sessions.

7. In the **Client ID** field, enter the unique identifier (client ID) that was assigned when you created the OAuth relying party (RP) client within the token provider (for more information, see Configuring OAuth clients in the PingFederate documentation).
8. Select a **Client Credentials Type**. This is required when configuring the **Code** login type.

   Choose from:

   - **Secret**
   - **Mutual TLS**
   - **Private Key JWT**

   > **Note**
   >
   > The OAuth client you use with PingAccess web sessions must have an OpenID Connect (OIDC) policy specified (for more information see Configuring OpenID Connect Policies.

9. Provide the information required for the selected credential type.

   Choose from:

   - **Secret** – Enter the **Client Secret** assigned when you created the OAuth relying party client in the token provider.
   - **Mutual TLS** – Select a configured **Key Pair** to use for Mutual TLS client authentication.
   - **Private Key JWT** – No additional information is required.

10. In the **Idle Timeout** field, specify 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).

    > **Note**
    >
    > If there is an existing valid PingFederate session for the user, an idle timeout of the PingAccess session might result in its re-establishment without forcing the user to sign on again.

11. In the **Max Timeout** field, specify the amount of time, in minutes, that the PA token remains active before expiring (the default is `240` minutes).
12. Click **Save**.

Configuring a rule

About this task

Rules are used to control the circumstances under which users can access the protected web server. Rules can grant or deny access based on criteria such as user parameters from the token provider, header values, network ranges, or web session attributes. You can configure any number of rules in your environment.

You can combine rules into rule sets, which combine multiple rules. You can configure rule sets to allow access to a resource if at least one rule's criteria is met, or to only allow access if all rules have their criteria met. Access control rules are processed before processing rules. Each type of rule is otherwise processed in the order you specify when you create the rule set.

You can further combine rule sets into rule set groups, which combine multiple rule sets. As with rule sets, rule set groups can allow access if any one rule set's criteria are met, or only if all rule sets' criteria are met. Rule sets are processed in the order you specify when you create the rule set group.

This example uses an HTTP request header rule to demonstrate how rules are created and used. Each environment has different requirements, and you can use any of the rules explained in the Rule management section according to your needs.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. From the **Type** menu, select **HTTP Request Header**.
5. In the **Field** column, in the **Header** field, enter the HTTP header name you want to match in order to grant or not grant the client access.
6. In the **Value** field, enter the values 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 in the **Value** field, or add a wildcard after the host name (`host*` or `host:*`) to match what is in the HTTP request.

7. If you need additional header pairs, click **Add Row** to add an additional row, then repeat steps 5-6.
8. Click **Save**.

Configuring an identity mapping

About this task

A header identity mapping can expose one or more attribute values to the protected application in HTTP request headers.

For more information about this procedure, including optional steps that are not included here, see Creating header identity mappings.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the mapping.
4. From the **Type** list, select **Header Identity Mapping**.
5. In the **Attribute to Header Mapping** section, in the **Attribute Name** field, enter the name of the attribute to retrieve from the user web session. For example, `sub` .
6. In the **Header Name** field, enter the name of the HTTP requests header to contain the attribute value.

> **Note**
>
> The 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. In the **Certificate to Header Mapping** section, enter the header name included in 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.

8. Click **Save**.

Configuring an application

About this task

The application represents the protected web application as a whole. By including the virtual host and the agent, it allows PingAccess to match incoming traffic to the application with the agent managing that application. By including a web session, you specify how user credential data is stored and for how long. After you create the application and its root resource, you can add one or more rules to control access to the protected web application.

Within the application, you can add one or more resources. Resources are specific components that require a different degree of security. You can apply different rules to a resource, letting you apply specific controls to portions of an application. This example procedure does not include resources. For information about adding resources to an application, see Configuring resource ordering in PingAccess.

For more information about this procedure, including optional steps that are not included here, see Adding an application.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click **+ Add Application**.
3. In the **Name** field, enter a unique name for the application, up to 64 characters, including special characters and spaces.
4. In the **Context Root** field, enter the context root for the application. This represents the context at which the application is accessed at the site.

   The context root 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.

5. From the **Virtual Host** list, select the virtual host you created.
6. In the **Application Type** section, select **Web**.

7. From the **Web Session** list, select the web session you created.
8. In the **Destination** section, select **Agent**, then select the agent that is installed on the same web server as the web application.
9. Click **Save**.
10. Click **Applications**, then go to **Applications > Applications**.
11. Expand the new application in the list and click the pencil icon .
12. Click the **Web Policy** tab.
13. Drag your rule from **Available Rules** onto the policy bar.
14. Click **Save**.

Protecting an API with PingAccess in an agent deployment

You can protect an application programming interface (API) from unwanted access using PingAccess.

Prerequisites

Before configuring your PingAccess deployment to protect an API:

- PingAccess must be installed and running. You can find the full procedure in Installing and Uninstalling PingAccess.
- You must have a configured token provider. The procedures vary depending on the token provider. Learn more in:
  - PingFederate.
  - PingOne.
  - Common token provider.
- You must have installed an agent on the web server or servers that host the API you want to protect. Learn more in:
  - PingAccessAgent for Apache (RHEL)
  - PingAccessAgent for Apache (SLES)
  - PingAccessAgent for Apache (Windows)
  - PingAccessAgent for IIS
  - PingAccessAgent for NGINX

Steps

Complete the following steps to protect your API:

1. Configure a virtual host.

   A virtual host represents the API you'll protect and contains information about its location.

2. Configure a rule.

   Rules determine who can access what content and under which circumstances.

3. Configure an identity mapping.

   An identity mapping enables you to pass identity information to the protected application using headers.

4. Configure an application.

   An application joins the other pieces together, giving users access to the API according to the configured rules.

5. Configure a resource.

   A resource specifies an API endpoint and the methods that can be used to access it.

Configuring a virtual host

About this task

The virtual host is the external-facing portion of an API. In an agent deployment, the virtual host contains the actual host name and port for the protected API.

For more information about this procedure, including optional steps that are not included here, see Creating new virtual hosts.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click **+ Add Virtual Host**.
3. In the **Host** field, enter the name for the virtual host.

   This is the host name of the protected API. For example, myHost.com. You can use a wildcard (`*`) for part or all of the host name. For example, `*.example.com` matches all host names ending in .example.com, and `*` matches all host names.

4. In the **Port** field, enter the port number for the virtual host. For example, `443` .
5. Click **Save**.

Configuring a rule

About this task

Rules are used to control the circumstances under which users can access the protected API. Rules can grant or deny access based on criteria such as user parameters from the token provider, header values, network ranges, or web session attributes. You can configure any number of rules in your environment.

You can combine rules into rule sets, which combine multiple rules. You can configure rule sets to allow access to a resource if at least one rule's criteria is met, or to only allow access if all rules have their criteria met. Access control rules are processed before processing rules. Each type of rule is otherwise processed in the order you specify when you create the rule set.

You can further combine rule sets into rule set groups, which combine multiple rule sets. As with rule sets, rule set groups can allow access if any one rule set's criteria are met, or only if all rule sets' criteria are met. Rule sets are processed in the order you specify when you create the rule set group.

This example uses an HTTP request header rule to demonstrate how rules are created and used. Each environment has different requirements, and you can use any of the rules explained in the Rule management section according to your needs.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. From the **Type** menu, select **HTTP Request Header**.
5. In the **Field** column, in the **Header** field, enter the HTTP header name you want to match in order to grant or not grant the client access.
6. In the **Value** field, enter the values 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 in the **Value** field, or add a wildcard after the host name (`host*` or `host:*`) to match what is in the HTTP request.

7. If you need additional header pairs, click **Add Row** to add an additional row, then repeat steps 5-6.
8. Click **Save**.

Configuring an identity mapping

About this task

A header identity mapping can expose one or more attribute values to the protected API in HTTP request headers.

For more information about this procedure, including optional steps that are not included here, see Creating header identity mappings.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the mapping.
4. From the **Type** list, select **Header Identity Mapping**.
5. In the **Attribute to Header Mapping** section, in the **Attribute Name** field, enter the name of the attribute to retrieve from the user web session. For example, `sub`.
6. In the **Header Name** field, enter the name of the header to contain the attribute value.

> **Note**
>
> 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. In the **Certificate to Header Mapping** section, enter the header name included in 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.

8. Click **Save**.

Configuring an application

About this task

The application represents the protected API as a whole. By including the virtual host and the agent, it allows PingAccess to route requests directed at the front-end name to the correct back-end resource. After you create the application and its root resource, you can add one or more rules to control access to the protected API.

Within the application, you can add one or more resources. Resources are specific components that require a different degree of security. You can apply different rules to a resource, letting you apply specific controls to portions of an application. This example procedure does not include resources. For information about adding resources to an application, see Configuring resource ordering in PingAccess.

For more information about this procedure, including optional steps that are not included here, see Adding an application.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click **+ Add Application**.
3. In the **Name** field, enter a unique name for the application, up to 64 characters, including special characters and spaces.
4. In the **Context Root** field, enter the context root for the API. This represents the context at which all of the API endpoints are accessed at the site.

   The context root 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.

5. From the **Virtual Host** list, select the virtual host you created.
6. In the **Application Type** section, select **API**.
7. In the **Destination** section, select **Agent**, then select the agent that is installed on the same web server as the API.

8. Click **Save**.
9. Click **Applications**, then go to **Applications > Applications**.
10. Expand the new application in the list and click the pencil icon .
11. Click the **Web Policy** tab.
12. Drag your rule from **Available Rules** onto the policy bar.
13. Click **Save**.

Configuring a resource

About this task

An application resource is a component within an application that requires a different level of security. By configuring resources for your API endpoints, you can add different rules for different endpoints and specify which methods are allowed.

For more information about this procedure, including optional steps that are not included here, see Adding application resources.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click to expand the application you created in the previous procedure.
3. Click the **Pencil** icon.
4. Click the **Resources** tab.
5. Click **+ Add Resource**.
6. In the **Name** field, enter a unique name up to 64 characters, including special characters and spaces.
7. In the **Path Patterns** field, enter a list of Uniform Resource Locator (URL) path patterns, within the context root, that identify this resource.

   > **Note**
   >
   > The path pattern must start with a forward slash (/). It begins after the application context root and extends to the end of the URL.
   > When automatic path pattern evaluation ordering is in use (default), patterns can contain one or more wildcard characters (`*`). No use of wildcards is assumed. For example, there is a difference between `/app/` and `/app/*` . If a request matches more than one resource, the most specific match is used.

8. In the **Resource Authentication** section, select **Standard**, using the same authentication for the resource as for the root application.
9. In the **Methods** field, enter the methods supported by the resource.

   Leave the asterisk default if the resource supports all HTTP methods, including custom methods.

> **Tip**
>
> Defining methods for a resource allows more fine-grained access control policies on API resources. If you have a server optimized for writing data (POST, PUT) and a server optimized for reading data (GET), you might want to segment traffic based on the operation being performed.

10. To log information about the transaction to the audit store, select the **Audit** check box.
11. To enable the resource, select the **Enabled** check box.
12. Click **Save**.

Protecting an API with PingAccess in a sideband deployment

You can protect an application programming interface (API) from unwanted access using PingAccess.

Prerequisites

Before configuring your PingAccess deployment to protect an API:

- PingAccess must be installed and running. You can find the full procedure in Installing and Uninstalling PingAccess.
- You must have a configured token provider. The procedures vary depending on the token provider. Learn more in:
  - PingFederate.
  - PingOne.
  - Common token provider.
- You must have installed a sideband client on the API gateway that serves the API you want to protect. To learn more, contact Ping professional services.

Steps

Complete the following steps to protect your API:

1. Configure a virtual host.

   A virtual host represents the API you'll protect and contains information about its location.

2. Configure a rule.

   Rules determine who can access what content and under which circumstances.

3. Configure an identity mapping.

   An identity mapping enables you to pass identity information to the protected application using headers.

4. Configure an application.

   An application joins the other pieces together, giving users access to the API according to the configured rules.

5. Configure a resource.

A resource specifies an API endpoint and the methods that can be used to access it.

Configuring a virtual host

About this task

The virtual host is the external-facing portion of an API. In a sideband deployment, the virtual host contains the actual host name and port for the protected API.

For more information about this procedure, including optional steps that are not included here, see Creating new virtual hosts.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click **+ Add Virtual Host**.
3. In the **Host** field, enter the name for the virtual host.

   This is the host name of the protected API. For example, myHost.com. You can use a wildcard (`*`) for part or all of the host name. For example, `*.example.com` matches all host names ending in .example.com, and `*` matches all host names.
4. In the **Port** field, enter the port number for the virtual host. For example, `443` .
5. Click **Save**.

Configuring a rule

About this task

Rules are used to control the circumstances under which users can access the protected API. Rules can grant or deny access based on criteria such as user parameters from the token provider, header values, network ranges, or web session attributes. You can configure any number of rules in your environment.

You can combine rules into rule sets, which combine multiple rules. You can configure rule sets to allow access to a resource if at least one rule's criteria is met, or to only allow access if all rules have their criteria met. Access control rules are processed before processing rules. Each type of rule is otherwise processed in the order you specify when you create the rule set.

You can further combine rule sets into rule set groups, which combine multiple rule sets. As with rule sets, rule set groups can allow access if any one rule set's criteria are met, or only if all rule sets' criteria are met. Rule sets are processed in the order you specify when you create the rule set group.

This example uses an HTTP request header rule to demonstrate how rules are created and used. Each environment has different requirements, and you can use any of the rules explained in the Rule management section according to your needs.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. From the **Type** menu, select **HTTP Request Header**.

5. In the **Field** column, in the **Header** field, enter the HTTP header name you want to match in order to grant or not grant the client access.
6. In the **Value** field, enter the values 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 in the **Value** field, or add a wildcard after the host name (`host*` or `host:*`) to match what is in the HTTP request.

7. If you need additional header pairs, click **Add Row** to add an additional row, then repeat steps 5-6.
8. Click **Save**.

Configuring an identity mapping

About this task

A header identity mapping can expose one or more attribute values to the protected API in HTTP request headers.

For more information about this procedure, including optional steps that are not included here, see Creating header identity mappings.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the mapping.
4. From the **Type** list, select **Header Identity Mapping**.
5. In the **Attribute to Header Mapping** section, in the **Attribute Name** field, enter the name of the attribute to retrieve from the user web session. For example, `sub` .
6. In the **Header Name** field, enter the name of the header to contain the attribute value.

> **Note**
>
> 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. In the **Certificate to Header Mapping** section, enter the header name included in 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.
8. Click **Save**.

Configuring an application

About this task

The application represents the protected API as a whole, including the virtual host, sideband client, and rules. By including the virtual host and sideband client, it allows PingAccess to route requests directed at the front-end name to the correct back-end resource. After you create the application and its root resource, you can add one or more rules to control access to the protected API.

Within the application, you can add one or more resources. Resources are specific components that require a different degree of security. You can apply different rules to a resource, letting you apply specific controls to portions of an application. This example procedure does not include resources. For information about adding resources to an application, see Configuring resource ordering in PingAccess.

For more information about this procedure, including optional steps that are not included here, see Adding an application.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click **+ Add Application**.
3. In the **Name** field, enter a unique name for the application, up to 64 characters, including special characters and spaces.
4. In the **Context Root** field, enter the context root for the API. This represents the context at which all of the API endpoints are accessed at the site.

   The context root 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.
5. From the **Virtual Host** list, select the virtual host you created.
6. In the **Application Type** section, select **API**.
7. In the **Destination** section, select **Sideband**, then select the client that is installed on the API gateway.
8. Click **Save**.
9. Click **Applications**, then go to **Applications > Applications**.
10. Expand the new application in the list and click the pencil icon .
11. Click the **Web Policy** tab.
12. Drag your rule from **Available Rules** onto the policy bar.
13. Click **Save**.

Configuring a resource

About this task

An application resource is a component within an application that requires a different level of security. By configuring resources for your API endpoints, you can add different rules for different endpoints and specify which methods are allowed.

For more information about this procedure, including optional steps that are not included here, see Adding application resources.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click to expand the application you created in the previous procedure.
3. Click the **Pencil** icon.
4. Click the **Resources** tab.
5. Click **+ Add Resource**.
6. In the **Name** field, enter a unique name up to 64 characters, including special characters and spaces.
7. In the **Path Patterns** field, enter a list of Uniform Resource Locator (URL) path patterns, within the context root, that identify this resource.

   > **Note**
   >
   > The path pattern must start with a forward slash (/). It begins after the application context root and extends to the end of the URL.
   > When automatic path pattern evaluation ordering is in use (default), patterns can contain one or more wildcard characters (`*`). No use of wildcards is assumed. For example, there is a difference between `/app/` and `/app/*` . If a request matches more than one resource, the most specific match is used.

8. In the **Resource Authentication** section, select **Standard**, using the same authentication for the resource as for the root application.
9. In the **Methods** field, enter the methods supported by the resource.

   Leave the asterisk default if the resource supports all HTTP methods, including custom methods.

   > **Tip**
   >
   > Defining methods for a resource allows more fine-grained access control policies on API resources. If you have a server optimized for writing data (POST, PUT) and a server optimized for reading data (GET), you might want to segment traffic based on the operation being performed.

10. To log information about the transaction to the audit store, select the **Audit** check box.
11. To enable the resource, select the **Enabled** check box.
12. Click **Save**.

Introduction to PingAccess

PingAccess is an identity-enabled access management product that protects web applications and APIs by applying security policies to client requests.

PingAccess allows you to protect sites, APIs, and other resources using rules and other authentication criteria. Working in conjunction with a configured token provider, PingAccess integrates identity-based access management policies through a federated corporate identity store using open standards access protocols.

> **Note**
>
> Valid token providers include PingFederate and other common token providers with the OAuth 2.0 and OpenID Connect (OIDC) protocols. For more information, see System requirements.

To help you get the most from PingAccess, this document offers insights about the product, such as:

- What can I do with PingAccess?
- How does PingAccess work?
- What can I configure with PingAccess?
- How do I choose a deployment model?

As you learn about PingAccess's features and functions, review Configuring and Customizing PingAccess for instructions on how to configure them.

For a comprehensive set of instructions on using the PingAccess interface, see the PingAccess User Interface Reference Guide.

PingAccess for Azure AD Overview

PingAccess for Azure AD is a free version of PingAccess for users of Microsoft Entra ID (formerly Microsoft Azure AD) that allows you to protect up to 20 applications.

> **Important**
>
> The PingAccess for Azure AD program ends on December 31, 2025. To continue using PingAccess, you must upgrade to a commercial PingAccess license. Learn more in:
>
> - Manage license keys
> - View or upload a new license

The goal of this solution is to allow for greater control over access to legacy on-premise applications through the use of PingAccess identity mapping functionality.

Learn more about configuring PingAccess for Azure AD in PingAccess for Azure AD.

PingAccess for Azure AD requires a premium license for Microsoft Entra ID. Learn more about licensing in https://learn.microsoft.com/en-us/azure/active-directory/app-proxy/application-proxy-ping-access-publishing-guide in the Microsoft documentation.

This free version of PingAccess includes a limited feature set that's intended to support the basic requirements for application protection using this solution. Users of PingAccess for Azure AD can upgrade to a full license allowing the use of the full PingAccess feature set.

Important

When your PingAccess for Azure AD license expires, you won't be able to access the PingAccess administrative application programming interface (API) or 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.

Note

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 isn't supported.

The following table details the capabilities of PingAccess for Azure AD compared to a full version of PingAccess. These capabilities are available in both the PingAccess administrative console and administrative API.

Capability	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 JSON Web Token (JWT).
Create load balancing strategies	Yes	Limited to Header-Based and Round Robin.
Configure web sessions	Yes	Limited to web sessions with OpenID Connect (OIDC) sign-on type CODE.
Configure token provider	Yes	Limited to Microsoft Entra ID authentication source.
Export/Import configuration	Yes	Limited to configurations that include only the features permitted by your 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

Capability	PingAccess	PingAccess for Azure AD
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 and access management (IAM) that allows you to control access by specifying the conditions that users must meet to access protected application and API resources.

The following sections describe the methods that PingAccess uses to control access and perform system functions. For more information on how you can use PingAccess, see:

- Configuring and Customizing PingAccess
- Reference Guides
- PingAccess User Interface Reference Guide

The main functionality of PingAccess enables you to protect an application or application programming interface (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 the 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

You can configure PingAccess to use PingFederate as the token provider or to use a common token provider through the OAuth 2.0 or OpenID Connect (OIDC) protocols.

- For more information on how to configure a token provider in the PingAccess administrative console, see Token provider.
- For more information on how to set up a connection between a token provider and PingAccess, see Token Providers.

> **Note**
>
> This section of the documentation provides information on how to configure a few common token providers as the token provider for PingAccess, while the previous link includes information on how to set up PingAccess to connect with the token provider.

Configure administrator authentication

Allow administrators to authenticate with a simple username and password or configure them to authenticate using single sign-on (SSO) or an API in conjunction with PingFederate. For more information, see Admin authentication.

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. For more information, see Log configuration.

Configure single logout (SLO)

End PingAccess sessions easily when used in conjunction with PingFederate managed sessions or compatible third-party OIDC providers. For more information, see Configuring a PingFederate runtime or Configuring OpenID Connect token providers.

Create clusters

Deploy PingAccess in a clustered environment to provide higher scalability and availability for critical services. Place a load balancer in front of the cluster to distribute connections to the nodes in the cluster. For more information, see Clustering in PingAccess.

Customize PingAccess look and feel

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

Customize with SDKs

Customize development with SDKs to extend the functionality of the PingAccess server. For more information, see PingAccess Add-on SDK for Java.

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 Attribute Sharing Profile (XASP) 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. For more information, see Configuration file reference.

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.

The developers page contains additional resources for developing applications to work with PingAccess.

Tune performance

Optimize a wide variety of PingAccess components for maximum performance. For more information, see Performance tuning.

Upgrade an existing installation

Upgrade an existing installation using the installer or selectively manage the upgrade process with the PingAccess upgrade utility. For more information, see Installing and Uninstalling PingAccess.

Use APIs

Use the PingAccess APIs to provide a powerful configuration and management experience outside the PingAccess user interface. For more information, see Accessing the PingAccess administrative API.

How does PingAccess work?

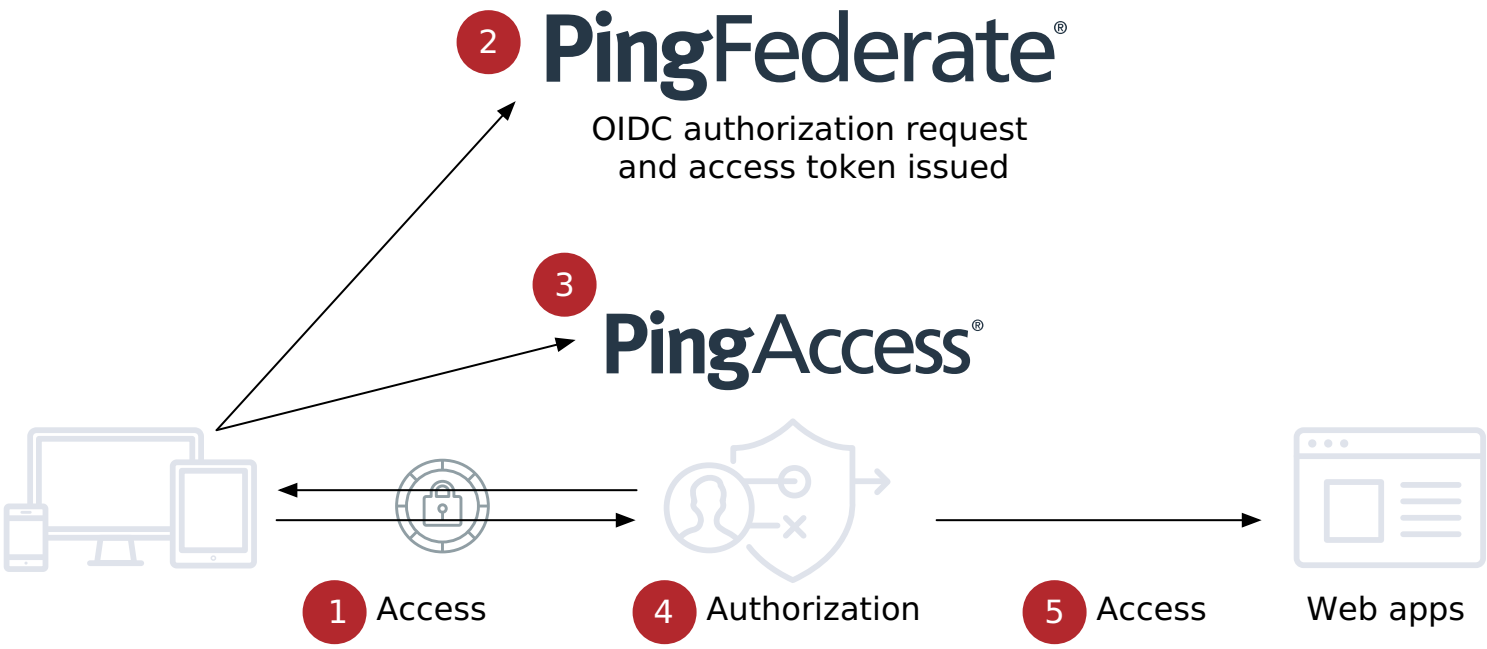
Access requests are either routed through a PingAccess gateway to the target site or intercepted at the target web application server by a PingAccess agent, which coordinates access policy decisions with a PingAccess policy server.

In either instance, PingAccess evaluates the policies applied to access requests for the target application and makes a policy-based decision to grant or deny access to the requested resource. After access is granted, PingAccess can modify client requests and server responses to provide additional identity information required by the target application.

WAM session initiation

When a user authenticates, PingAccess applies your configured application and resource-level policies to the Web Access Management (WAM) request.

After completing policy evaluation and determining that the authenticated user should be granted access to a site, PingAccess performs any required token mediation between the backend site and the authenticated user. PingAccess then grants the user access to the site.



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

> **Note**
>
> When using an OP, you must already have an OAuth client configured in PingAccess.
> - For information on configuring an OAuth client within PingFederate, see Configure PingFederate as the token provider for PingAccess and the Administrator's Reference Guide in the PingFederate documentation.
> - To configure the OAuth client within PingAccess, see Connect PingAccess to PingFederate.

3. The OP follows the appropriate authentication process, evaluates domain-level policies, and issues an OIDC ID token to PingAccess.
4. PingAccess validates the ID token and issues a PingAccess token and sends it to the browser in a cookie during a redirect to the original target resource.

After gaining access to the resource, PingAccess evaluates application and resource-level policies and can optionally audit the request.

> **Note**
>
> PingAccess can perform token mediation by exchanging the PingAccess token for the appropriate security token from the PingFederate Security Token Service (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 pictured).

> **Note**
>
> For more information, see the Session management configuration.

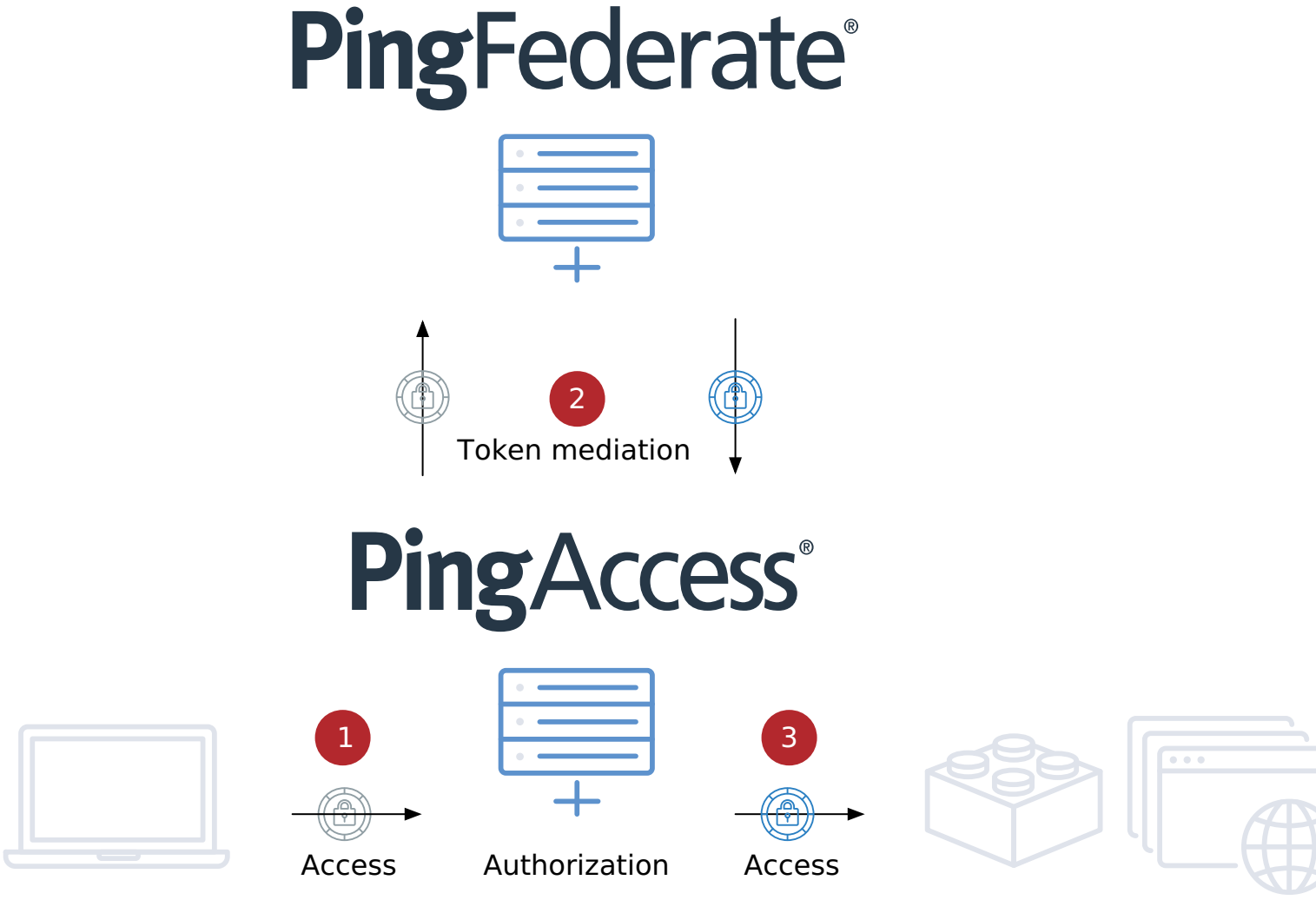
Token mediation

Token mediation allows a PingAccess gateway to use a PingFederate token generator to exchange the PingAccess token or an OAuth bearer token for a security token used by the foreign authentication system.

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

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. After token mediation, PingAccess caches the token used to access the application for continued use during the session.

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



1. A user requests a resource from PingAccess with a PingAccess token or OAuth bearer token.

> **Note**
>
> This example assumes the user has already obtained a PingAccess token or OAuth bearer token. For information on how users authenticate with PingFederate and obtain a PingAccess token or OAuth bearer token, see Session management configuration.

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 (step not pictured).

> **Note**
>
> You can't access a mediated token through a Groovy rule because token mediation occurs after PingAccess rule processing.

You can configure token mediation cache settings in the `run.properties` file using the following parameters:

pa.ehcache.ServiceTokenCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for token mediation. The default value 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 value is 1800.

pa.ehcache.ServiceTokenCache.timeToLiveSeconds

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

What can I configure with PingAccess?

PingAccess includes a wide range of features to customize your identity access management deployment.

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 through the PingAccess Agent Protocol (PAAP), which defines the possible interactions between agents and policy server.

Agents have a name to identify them and a shared secret for authentication with the 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 corresponding to a single target site. Applications also use a common web session and identity mapping.

To protect applications and their resources, you can apply access control and request processing rules on the policy manager page using the following options:

PingAccess gateway

In a gateway deployment, the target application is specified as a site.

PingAccess agent

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:

1. A user attempts to access a PingAccess web application configured with an authentication requirement list containing the values, such as password and certificate.
2. PingAccess redirects the user to PingFederate requesting either password or certificate user authentication.
3. PingFederate authenticates the user based on the password and issues an OpenID Connect (OIDC) ID token to PingAccess, containing the authentication method that was used.
4. PingAccess ensures that the authentication method matches the requirements and redirects the user to the originally requested application with the PingAccess cookie set.
5. 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 (certificate), they are redirected to PingFederate to authenticate with a certificate.

If you configure applications with authentication requirement lists that have no overlap, a user navigating between those applications might be required to authenticate each time they visit an application. So, when you're configuring authentication requirement lists to protect higher value applications with step-up authentication, consider including stronger forms of authentication on lower value applications as well.

Auth token management

Auth token management settings define the issuer and signing configuration used by JSON Web Token (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 causes the first listed target in the site configuration to be used unless it fails. Secondary targets are only 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 can be issued by a certificate authority (CA) or self-signed. Use CA-issued certificates 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

To provide higher scalability and availability for critical services, configure PingAccess in a clustered environment.

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.

> **Note**
>
> You can configure any number of clustered engines in a cluster, but you can only configure one administrative console and one replica administrative console in a cluster.

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 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 are 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 does not select a target that is in a failed state.

Policies

Policies are rules, rule sets, or groups of rule sets applied to an application and its resources. Policies define how and when a client can access target sites. 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
- Applying them to applications and resources

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 that 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 can 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 and 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. You can configure multiple 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.

How do I choose a deployment model?

PingAccess supports three deployment models.

The gateway, agent, and sideband deployment models each have advantages and disadvantages. Review them before selecting a model for your environment.

Gateway model

What is the gateway model?

In the gateway model, traffic is initially directed to a PingAccess node, and PingAccess grants or denies access directly. The application in PingAccess is configured with the site as the destination.

Pros

- Less cross-team coordination required — You can implement and maintain a gateway deployment with less coordination with application teams because the PingAccess infrastructure is installed on separate systems from the web servers.
- Simpler setup — Because the PingAccess nodes are the only required components, this deployment model can be set up more quickly than the other models.
- Simpler upgrade — The only components you must upgrade in a gateway deployment are the PingAccess nodes.

> **Tip**
>
> You can upgrade PingAccess with zero downtime in a clustered environment.

- Simpler troubleshooting — Issues are easier to isolate because there are fewer components sharing a system with the PingAccess infrastructure.
- Simpler logging — All transactions that PingAccess processes are audited by the engine node, making it easier to view logs for a specific event.

Cons

- Network impact — Using the gateway deployment model requires that you restructure your existing network to route traffic through PingAccess.
- Additional network overhead — The overhead of an additional network hop can theoretically exceed a latency budget. This rarely happens in practice, and the agent model often makes a similar addition to latency, but this might occur in some environments.

Agent model

What is the agent model?

In the agent model, traffic is directed to the application, which has an agent plugin installed on the web server. The agent grants or denies access, and queries the PingAccess node when it requires additional information. The application in PingAccess is configured with the agent as the destination.

Pros

- No network changes — Because the PingAccess agents are installed on the web servers, no network changes are required.
- Minor performance improvements — In some cases, the agent can determine whether to grant access using cached data, which can reduce latency. In most cases, though, the agent must communicate with a PingAccess node, which results in latency similar to the gateway model.

Cons

- Greater maintenance effort — You must maintain and upgrade agents independently on each web server.
- Unavailable features — Some features can't be used in an agent deployment. You can't:
  - Rewrite the request URL
  - Rewrite response headers
  - Rewrite request or response body content
- Cross-team coordination — Because the agents are installed directly on the web server, you might have to coordinate with other teams to install and maintain them.
- Complex tracking — Keeping track of all agents can be difficult.
- Difficult troubleshooting — Because the agent model involves more systems which can be varied in their OS and web server versions, troubleshooting issues can be more difficult.
- Version dependencies — Because the agent must be installed as a plugin on the web server, there are dependencies on the web server and OS versions that aren't present in the gateway model.
- Less centralized logging — To view the logging for a specific transaction, you must review the agent audit log and the web server access logs.

Sideband model

What is the sideband model?

In the sideband model, traffic is directed to an application programming interface (API) gateway, such as Apigee, Kong, or Mulesoft. The API gateway makes a backchannel call to PingAccess to determine if it should grant or deny the request and determine what modifications should be made to the request or response. The application in PingAccess is configured with the sideband client as the destination.

Pros

- No network changes — Because the PingAccess sideband clients are installed on the API gateways, no network changes are required.
- Simpler troubleshooting — Issues are easier to isolate because there are fewer components sharing a system with the PingAccess infrastructure.
- Simpler logging — All transactions processed by PingAccess are audited by the engine node, making it easier to view logs for a specific event.

Cons

- Greater maintenance effort — You must maintain and upgrade API gateway integration kits independently on each API gateway.
- Unavailable features — Some features may not be available depending on your API gateway.

- Additional network overhead — The overhead of an additional network hop can theoretically exceed a latency budget. This rarely happens in practice, and the gateway and agent models often make a similar addition to latency, but this might occur in some environments.
- Cross-team coordination — Because the sideband clients are installed directly on the API gateways, you might have to coordinate with other teams to install and maintain them.
- Complex tracking — A sideband deployment has more components to maintain than a gateway deployment.
- Version dependencies — Because the sideband client must be installed on the API gateway, there are dependencies on the API gateway and operating system versions that are not present in the gateway model.

Installing and Uninstalling PingAccess

Use this section for instructions on installing, configuring, and starting PingAccess.

- Before installing PingAccess, review:
  - Installation requirements.
- Install PingAccess:
  - On Linux.
  - On Windows.
- After installing PingAccess, you can:
  - Start PingAccess.
  - Access the admin console for the first time.
  - Access the PingAccess administrative API.
  - Change configuration database passwords.
- To stop, run, or uninstall PingAccess, see:
  - Stopping PingAccess.
  - Running PingAccess as a service.
  - Uninstalling PingAccess.

> **Note**
>
> These features are affected by the settings in the configuration file. For more information, see the Configuration file reference.

Installation requirements

Before you install PingAccess, review the following system, hardware, and port requirements.

System requirements

Make sure that your system meets the following requirements for PingAccess deployment and configuration.

Ping Identity qualifies the following configurations and certifies that they are compatible with the product. Variations of these platforms, such as differences in operating system version or service pack, are supported until the platform or other required software creates potential conflicts.

> **Note**
>
> PingAccess currently supports IPv4 addressing but not IPv6 addressing.

System component	Requirements
Operating systems	• Amazon Linux 2 ◦ Amazon Linux 2022 ◦ Amazon Linux 2023 • Canonical Ubuntu 20.04 (LTS) ◦ Canonical Ubuntu 22.04 (LTS) • Microsoft Windows Server 2016 (x64) ◦ Microsoft Windows Server 2019 (x64) ◦ Microsoft Windows Server 2022 (x64) • Oracle Linux 7.9 (Red Hat Compatible Kernel) ◦ Oracle Linux 8.6 (Red Hat Compatible Kernel) • Red Hat Enterprise Linux ES 7.9 ◦ Red Hat Enterprise Linux ES 8.8 ◦ Red Hat Enterprise Linux ES 9.2 • SUSE Linux Enterprise Server 12 SP5 ◦ SUSE Linux Enterprise Server 15 SP4 **Note** PingAccess was tested with default configurations of operating system components. If your organization has custom implementations or has installed third-party plug-ins, PingAccess server deployment might be affected.
Docker support	***Docker version*** 20.10.17 You can find the PingAccess Docker image on DockerHub and more information in Ping Identity's DevOps documentation. **Note** Only the PingAccess software is licensed under Ping Identity's end user license agreement. Any other software components contained within the image are licensed solely under the terms of the applicable open source or third-party license. Ping Identity accepts no responsibility for the performance of any specific virtualization software and in no way guarantees the performance or interoperability of any virtualization software with its products.

System component	Requirements
Virtual systems	Although Ping Identity doesn't qualify or recommend any specific virtual machine (VM) products, PingAccess runs well on several, including: • VMWare • Xen • Windows Hyper-V. **Note** This list of products is provided only as an example. We view all products in this category equally. Ping Identity accepts no responsibility for the performance of any specific virtualization software and does not guarantee the performance or interoperability of any VM software with its products.
Java environments	• Amazon Corretto 11 (64-bit) ◦ Amazon Corretto 17 (64-bit) ◦ Amazon Corretto 21 (64-bit) • OpenJDK 11 (64-bit) ◦ OpenJDK 17 (64-bit) ◦ OpenJDK 21 (64-bit) • Oracle Java SE Development Kit (JDK) 11 (64-bit) ◦ Oracle Java SE Development Kit (JDK) 17 (64-bit) ◦ Oracle Java SE Development Kit (JDK) 21 (64-bit) **Note** The Ping Identity Java support policy applies. **Important** Ping Identity intends to remove Java 11 support from PingAccess in December 2025.
PingFederate	The following versions of PingFederate are fully certified with this version of PingAccess: • PingFederate 11.3 • PingFederate 12.0 • PingFederate 12.1 • PingFederate 12.2 Other versions of PingFederate are expected to be compatible with this version of PingAccess as described in Ping Identity's end of life policy. **Note** Some features rely on a specific version of PingFederate to work. This will always be noted in the feature's description.

System component	Requirements

End-user browsers	• Google Chrome
	  ◦ Google Android (Chrome)
	• Microsoft Edge
	• Mozilla Firefox
	• Apple Safari
	  ◦ Apple iOS (Safari)

Admin console browsers	• Google Chrome • Microsoft Edge • Mozilla Firefox

Audit event storage (external database)	• MS SQL Server 2019
	  ◦ MS SQL Server 2022
	• Oracle 19c
	• PostgreSQL 13
	  ◦ PostGreSQL 16

Hardware security module	You can find more information about configuring a hardware security module (HSM) in Hardware security module providers. PingAccess certifies the following HSMs: ***AWS CloudHSM 5.9.0*** **Note** AWS CloudHSM is supported with JDK 11. If you plan to use AWS CloudHSM, you must also deploy your environment on a Linux or Windows operating system that is compatible with both PingAccess and AWS CloudHSM. ***Thales Luna Cloud HSM Services and Luna Network HSM (Luna HSM Client 10.x)*** **Note** PingAccess requires you to use Java 8 with Luna HSMs. This is the only exception to PingAccess's removal of Java 8 support.
Supported HTTP versions	HTTP 1.1

<table>
<tr><th>System component</th><th>Requirements</th></tr>
<tr><td>OpenID Connect (OIDC) providers</td><td>Ping Identity strives to support any third-party OIDC-compliant provider. The following table includes some of the most common providers used with PingAccess:
<table>
<tr><th>Provider</th><th>Provider Type</th></tr>
<tr><td>PingFederate</td><td>PingFederate</td></tr>
<tr><td>PingOne for Enterprise</td><td>Common</td></tr>
<tr><td>PingOne SSO</td><td>PingOne</td></tr>
<tr><td>PingOne Advanced Identity Cloud</td><td>PingOne Advanced Identity Cloud</td></tr>
<tr><td>PingAM</td><td>PingAM</td></tr>
<tr><td>Azure</td><td>Common</td></tr>
<tr><td>Okta</td><td>Common</td></tr>
</table>
</td></tr>
</table>

Hardware requirements

> **Note**
>
> Although it's 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 vary, run 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

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

Note

Direction refers to the direction of requests relative to PingAccess:

Inbound requests
Requests that PingAccess receives from external components.

Outbound requests
Requests that PingAccess sends to external components.

Service	Port details	Source	Description
PingAccess administrative console	***Protocol*** HTTPS ***Transport*** TCP ***Default port*** 9000 ***Destination*** PingAccess admin console ***Direction*** Inbound	PingAccess administrator browser, PingAccess administrative application programming interface (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. Learn more in the Configuration file reference guide. **Note** 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 admin console ***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. Learn more in the Configuration file reference guide. **Note** This port is also used by clustered engine nodes and the replica admin node to pull configuration data using the admin REST API.

Service	Port details	Source	Description
PingAccess engine	***Protocol*** HTTP or HTTPS ***Transport*** TCP ***Default port*** 3000* **Note** Any additional engine listener ports defined in the configuration must be open as well. ***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. Learn more in the PingAccess user interface reference guide.
PingAccess agent	***Protocol*** HTTP or 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. Learn more in the Configuration file reference guide.
PingAccess sideband (optional)	***Protocol*** HTTP or HTTPS ***Transport*** TCP ***Default port*** 3020 ***Destination*** PingAccess engine ***Direction*** Inbound	Sideband client (an API gateway such as Kong Gateway or Apigee)	Used for incoming sideband requests to the PingAccess runtime engine. Configurable using the `sideband.http.port` property of the `run.properties` file. Learn more in the Configuation file reference guide. **Note** The default value of the `sideband.http.enabled property` is `false`. This property must be set to `true` to configure a sideband client.

Service	Port details	Source	Description
PingFederate traffic	***Protocol*** HTTPS ***Transport*** TCP ***Default port*** 9031 ***Destination*** PingFederate ***Direction*** Outbound	PingAccess engine	Used to validate OAuth access token and ID tokens, make Security Token Service (STS) calls for identity mediation, and return authorized information about a user. Configurable using the `PingFederate Settings` page within PingAccess. Learn more in the PingAccess user interface reference guide.

Installing PingAccess on your system

Install PingAccess on Linux, on Windows through an installation wizard, or on Windows through the command-line interface (CLI).

Before you begin

- Ensure you've met the installation requirements.
- Ensure you're signed on to your system with appropriate privileges to install and run an application.

> **Note**
>
> On Linux, install and run PingAccess as a non-root user.

- Install a supported Java runtime.
- The system or user environment variable `JAVA_HOME` must exist and be set to a value that represents the location of your Java installation, such as `usr/java/jdk 1.8.0_74`.
- Add the relevant Java directory path to the `PATH` variable so it's available for scripts that depend on it:
  - On Linux: Add the Java Runtime Environment (JRE) `/bin` directory path (for example, `usr/lib64/jvm/jre/bin`).
  - On Windows installer: Add the `javapath` directory path (for example, `C:\Program Files\Oracle\Java\javapath`).
  - On Windows CLI: Add the `javapath` directory path (for example, `C:\Program Files\Oracle\Java\javapath`).
- You must have a `pingaccess.lic` license file.

> **Note**
>
> If you do not have a PingAccess license, you can request an evaluation key at https://support.pingidentity.com/s/. During the first run of PingAccess, you will be prompted to upload the license file.
> If you are using an existing configuration file to configure the system, copy the configuration file to the system and rename it `data.json`. For more information about exporting the configuration from an existing system, see Exporting PingAccess configurations.

Linux

Installing PingAccess on Linux
About this task

To install PingAccess on a Linux system:

Steps

1. Download the distribution `.zip` archive from the PingAccess downloads page.
2. Extract the distribution `.zip` archive into your installation directory.
3. **Optional:** If you are using an existing configuration file to configure the system, move the `data.json` file to the `<PA_Home>/data/start-up-deployer` directory.

   > **Note**
   >
   > When you start PingAccess for the first time, if this configuration is present it will be imported. After a successful import, the `data.json` file is deleted. If the configuration is present but cannot be imported, PingAccess is not started.

   > **Tip**
   >
   > If you're deploying PingAccess in a cluster configuration, see the configuration documentation.

Next steps

Access the administrative console to complete the configuration.

Windows installer

Installing PingAccess on Windows using the installer
About this task

To install PingAccess on a Windows system using the installer:

Steps

1. Download the PingAccess Windows installer from the PingAccess downloads page.
2. Double-click on the installer icon to launch the PingAccess setup wizard.
3. Click **Next** and follow the prompts to complete the installation using the following information for your selected operational mode.

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: ◦ You must install and configure a clustered admin node. ◦ You must have a configuration data archive file available for the replica admin node. For more information, see Clustering in PingAccess in *Reference Guides*. **Note** Install the clustered replica admin node on a separate machine in the same network.
Clustered engine node	Ports: ◦ PingAccess agent protocol: TCP 3030 Prerequisites: ◦ You must install a clustered admin node. ◦ You must have a configuration data archive file available for the clustered engine node. For more information, see Clustering in PingAccess in *Reference Guides*.

4. Copy the Uniform Resource Locator (URL) of the PingAccess administrative console that is displayed on the final page 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 administrative console of the instance you have just installed.

Next steps

Access the administrative console to complete the configuration.

Windows CLI

Installing PingAccess on Windows from the command line
About this task

To install PingAccess on a Windows system from the CLI:

Steps

1. Download the distribution `.zip` archive.
2. Extract the distribution `.zip` archive into your installation directory.
3. **Optional:** If you are using an existing configuration file to configure the system, move the `data.json` file to the `<PA_Home>/data/start-up-deployer` directory.

> **Note**
> When you start PingAccess for the first time, if this configuration is present it will be imported. After a successful import, the `data.json` file is deleted. If the configuration is present but cannot be imported, PingAccess is not started.

Next steps

Access the administrative console to complete the configuration.

Starting PingAccess

After installing PingAccess, start the PingAccess service.

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:

   Choose from:

   - On Linux: `cd <PA_HOME>/bin`

- On Windows: `cd <PA_HOME>\bin`

2. Start the `run` script for the platform:

   Choose from:

   - On Linux: `./run.sh`
   - On Windows: `run.bat`

Result:

PingAccess starts when you see the message `PingAccess running…` in the command window.

Accessing the administrative console for the first time

After installing and starting PingAccess, access the administrative console and perform configuration and first-time sign on 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.

   > **Note**
   >
   > If you haven't yet installed a PingAccess license, the server redirects you to the **License Upload** window outside of the main UI. For more information, see the PingAccess User Interface Reference Guide.

2. Sign on 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, and then click **Continue**.

   > **Note**
   >
   > The new password must conform to the rules specified by the `pa.admin.user.password.regex` property in `run.properties`. For more information about these properties, see the Configuration file reference.

   Result:

   The PingAccess administrative console opens.

Result

After successfully signing on, PingAccess creates a backup of the current configuration to allow the administrator to revert any changes made, stored in `<PA_HOME>/data/archive`. You can control the number of backup files to keep using the `pa.backup.filesToKeep` property in the `run.properties` file. For more information about this property, see the Configuration file reference.

> **Caution**
>
> Because the backup file contains your complete PingAccess configuration, ensure the file is protected with appropriate security controls in place.

Accessing the PingAccess administrative API

Access the PingAccess administrative application programming interface (API).

Steps

- Send an HTTP request to this 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`.

Accessing the interactive administrative API documentation

View interactive documentation for the administrative API endpoints.

Steps

1. Launch your browser and go to `https://<host>:<admin-port>/pa-admin-api/v3/api-docs/`.

   Example:

   `https://localhost:9000/pa-admin-api/v3/api-docs/`

   > **Note**
   >
   > The browser might prompt you to enter your credentials.

2. Enter the administrator username and password.
3. Use the administrative API to perform a variety of administrative tasks, such as gathering information.

Example:

To use the interactive administrative API documentation to see all defined applications:

1. Click to expand the `/applications` endpoint.
2. Click to expand the `GET` method (`GET /applications`).
3. Enter values for the parameters or leave them all blank.
4. Click **Try It Out**.

Result:

The request Uniform Resource Locator (URL), response body, response code, and response headers display.

Changing configuration database passwords

Rotate the database passwords for the PingAccess configuration database.

About this task

The PingAccess configuration database is protected by randomly-generated passwords on startup. You can rotate these passwords for additional security.

Steps

1. Open a terminal window and go to the `<PA_HOME>/bin` directory.
2. To ensure the `JAVA_HOME` environment variable is set correctly, run the `echo $JAVA_HOME` command.
3. To ensure the proper Java executable is in your path, run the `java -version` command.

> **Note**
>
> If this command returns a value indicating that the Java executable is not a supported version, correct this issue before continuing.

4. Stop PingAccess.
5. Run the relevant rotation script for your environment:

   Choose from:

   - For Windows: `db-passwd-rotate.bat`
   - For Linux: `db-passwd-rotate.sh`

6. Restart PingAccess.

Stopping PingAccess

Stop PingAccess as a prerequisite for maintenance or uninstallation tasks.

Steps

- Stop PingAccess:

  Choose from:

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

Running 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 starts.

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

> **Tip**
>
> Before setting PingAccess up to run as a service, manually start the server to make sure that PingAccess runs normally. For more information, see Accessing the administrative console for the first time.

- To manage PingAccess as a service on a Linux operating system, see Managing the PingAccess Linux service.
- To manage PingAccess as a service on a Windows 64-bit operating system, see Managing the PingAccess Windows service.

Managing the PingAccess Linux service

Configure PingAccess to run as a Linux systemv or systemd service, or remove the PingAccess Linux service.

About this task

Configuring PingAccess to run as a Linux systemv or systemd service causes it to start automatically when Linux starts.

> **Note**
>
> To configure multiple instances of PingAccess as Linux services, see the Linux systemv tab.

Linux systemv

Configuring PingAccess to run as a Linux systemv service
About this task

> **Note**
> The service script will only start if *<JAVA_HOME>* and *<PA_HOME>* are set and if the script can find the PingAccess license file.

To configure multiple instances of PingAccess on a single host as Linux services, make the following modifications to the script for each service:

- Use a unique script name for each instance.
- Use a separate directory structure for each instance in the file system.
- Configure the following settings in the script file for each instance:

Setting	Description
<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 username used to run the service.

To configure PingAccess to run as a Linux systemv service:

Steps

1. Copy the PingAccess script file from the `<PA_HOME>/sbin/linux/pingaccess` directory to the `/etc/init.d` directory.
2. **Optional:** Create a new user to run PingAccess.
3. Create the `/var/run/pingaccess` directory.

   > **Note**
   > Ensure that the user who will run the service has read and write permissions to the folder.

4. Edit the `/etc/init.d/pingaccess` script file and set the values of the following variables at the beginning of the script:

Variable	Description
`export <JAVA_HOME>=`	Specify the Java install folder.
`export <PA_HOME>=`	Specify the PingAccess install folder.

Variable	Description
`export USER=` (Optional)	Specify a username to run the service or leave empty for the default.

5. To register the service, from the `/etc/init.d` directory, run:

```
chkconfig --add pingaccess
```

6. To make the service script executable, run:

```
chmod +x pingaccess
```

Next steps

After registering, 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 process identifier (PID).

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

Linux systemd

Configuring PingAccess to run as a Linux systemd service
About this task

> **Note**
>
> The service script will only start if *<JAVA_HOME>* and *<PA_HOME>* are set and if the script can find the PingAccess license file.

To configure PingAccess to run as a Linux systemd service:

Steps

1. Copy the configuration file from the `<PA_HOME>/sbin/linux/pingaccess.service` directory to the `/etc/systemd/system/pingaccess.service` directory.
2. In the `pingaccess.service` file, replace the following variables:
    - Replace *<${PA_HOME}>* with the path to the PingAccess instance.
    - Replace *<${PA_USER}>* with the username used to run the service.
    - Replace *<${PA_JAVA_HOME}>* with the path to the Java installation folder.
3. To allow read-write activity on the service, run:

```
chmod 644 /etc/systemd/system/pingaccess.service
```

4. To load the systemd service, run:

```
systemctl daemon-reload
```

5. To enable the service, run:

```
systemctl enable pingaccess.service
```

6. To start the service, run:

```
systemctl start pingaccess.service
```

Removing the PingAccess Linux service

About this task

> **Note**
>
> You must run the following commands as the `root` user.

To remove the PingAccess service from a Linux system:

Steps

1. To stop the service, run the **`/etc/init.d/pingaccess stop`** command.
2. Run the **`chkconfig --delete pingaccess`** command.
3. **Optional:** Delete the `/etc/init.d/pingaccess` script.

Managing the PingAccess Windows service

Configure PingAccess to run as a Windows service through an installer or the command line, or remove the PingAccess Windows service.

Before you begin

Install PingAccess and manually start the server to make sure that it runs normally. For more information, see Accessing the administrative console for the first time.

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

About this task

Configuring PingAccess as a Windows service causes it to start automatically when Windows starts. To configure PingAccess as a service on a Windows 64-bit operating system:

Steps

1. Ensure that you are signed on with full administrator privileges.
2. Select one of the following tabs to proceed.

Manual Installation

Configuring PingAccess to run as a Windows service
About this task

To configure PingAccess to run as a Windows service:

Steps

1. Open a command prompt as an administrator.
2. In the command prompt, go to the `<PA_HOME>\sbin\windows` directory and run the `install-service.bat` file.
3. In Windows, go to **Control Panel → Administrative Tools → Services**.
4. From the list of available services, right-click **PingAccess Service** and select **Start**.

   You can change the default **Start type** setting in the **Properties** dialog.

Result

The service starts immediately and restarts automatically on reboot.

From the Command Line

Configuring PingAccess to run as a Windows service from the command line
About this task

To configure PingAccess to run as a Windows service from the command line:

Steps

1. Go to the `<PA_HOME>\sbin\windows` directory and run the `install-service.bat` file.
2. To set the PingAccess service to start automatically, run `sc config PingAccess start= auto`.

Result

The service starts immediately and restarts automatically on reboot.

Removing the PingAccess Windows service

Before you begin

Make sure you have PingAccess administrator privileges.

About this task

To remove the PingAccess service from a Windows system:

Steps

1. Open a command prompt.
2. Stop the PingAccess service.
3. Go to the `<PA_HOME>\sbin\windows` directory.
4. Run the `uninstall-service.bat` file.
5. After the script runs, remove the *<PA_HOME>* environment variable from the system.

Uninstalling PingAccess

Uninstall PingAccess.

About this task

To uninstall PingAccess:

Steps

1. Stop PingAccess as appropriate for your system.
2. If you installed PingAccess using the Windows installer, use one of these options:

   Choose from:

   - Double-click on the installer icon, then click **Remove** and follow the prompts.
   - Open the Windows **Add or Remove Programs** tool, then locate and uninstall PingAccess.
3. If PingAccess is installed on a Linux system and has been configured to start automatically, remove the PingAccess service.
   1. Run the command: `chkconfig --delete pingaccess`
   2. **Optional:** Delete the `/etc/init.d/pingaccess` script.
4. Delete the PingAccess installation directory if it is present.

Backing up and restoring PingAccess

This section provides instructions for backing up and restoring PingAccess.

The tools in this section let you create backups of your PingAccess environment and restore your environment from them. You should back up your environment regularly.

- If you need disaster recovery, see Backing up and restoring PingAccess using a `.zip` archive.
- If you need to restore an environment's configuration or to test in a new environment, see Backing up and restoring PingAccess using a JSON file.

.zip archive">

Backing up and restoring PingAccess using a `.zip` archive

Use a `.zip` archive to back up and restore PingAccess.

Using a `.zip` archive is appropriate for disaster recovery because it uses automatically generated backups and restores the entire PingAccess configuration.

> **Note**
>
> The system on which you are restoring must have the same major and minor version of PingAccess installed before you begin the restoration.

.zip archive">

Backing up PingAccess using a `.zip` archive

Back up your PingAccess configuration by copying a `zip` archive and additional customized files to another system.

About this task

To back up your PingAccess configuration using a `.zip` archive for disaster recovery purposes:

Steps

1. In the PingAccess system, go to `<PA_HOME>/data/archive` and copy the most recent `.zip` archive to another system.

   These archives are automatically created when an administrative user authenticates to the administrative console. The maximum number of backups is specified by the `pa.backup.filesToKeep` property in the `run.properties` file.
2. **Optional:** If you have created or customized templates, copy the contents of the `<PA_HOME>/conf/template` directory to another system.
3. **Optional:** If you have created custom plugins, copy the contents of the `<PA_HOME>/deploy` directory to another system.
4. **Optional:** If you have created or customized localization, copy the contents of the `<PA_HOME>/conf/localization` directory to another system.

.zip archive">

Restoring PingAccess using a `.zip` archive

Restore your PingAccess configuration using a `.zip` archive and additional customized files.

About this task

This procedure restores the PingAccess configuration using a `.zip` archive and additional customization files. You can use this method for disaster recovery because it uses an automatically-generated file and restores the entire PingAccess configuration.

To restore your PingAccess configuration using a `.zip` archive and additional customized files:

Steps

1. If PingAccess is not installed on the system, install the same version of PingAccess used to create the backup.

   > **Note**
   >
   > The maintenance version does not have to be identical. For example, a backup made on PingAccess 6.3.0 could be restored on 6.3.1.

2. Stop PingAccess.
3. Extract the backup `.zip` archive to `<PA_HOME>`.
4. **Optional:** If you backed up custom templates, copy the backed up content to the `<PA_HOME>/conf/template` directory.
5. **Optional:** If you backed up custom plugins, copy the backed up content to the `<PA_HOME>/deploy` directory.
6. **Optional:** If you backed up custom localization, copy the backed up content to the `<PA_HOME>/conf/localization` directory.
7. Restart PingAccess.

   Result:

   Your PingAccess configuration is reverted to the configuration in the backup archive.

Backing up and restoring PingAccess using a JSON file

Use a JavaScript Object Notation (JSON) file to back up and restore PingAccess.

This method is appropriate for reverting to a prior configuration or testing a configuration in a new environment. It uses manually-generated backups and restores the PingAccess database configuration.

Backing up PingAccess using a JSON file

Back up your PingAccess configuration by copying a JavaScript Object Notation (JSON) file to another system.

About this task

Back up your PingAccess configuration by copying a JSON file to another system.

> **Note**
>
> Large PingAccess configurations can take upwards of 30 minutes to export. During an export, you cannot modify the PingAccess configuration.

Steps

1. Click **Settings**, then go to **System > Configuration Export/Import**.
2. Click **Export Configuration**.

   The downloaded file name is `pa-data-<timestamp>.json`.

   > **Note**
   >
   > The *<timestamp>* value is formatted `MM-DD-YYYY.hh.mm.ss`. For example, a date and time of January 31, 2020 1:35 PM would be encoded as `01-31-2020.13.35.00` in the file name.

3. Copy the generated file to another system.
4. If you plan to restore PingAccess in a new environment, copy the `<PA_Home>/conf/pa.jwk` file, and save it with the generated JSON file.

Restoring PingAccess using a JSON file

Restore your PingAccess configuration using a JavaScript Object Notation (JSON) file.

About this task

The **Import Configuration** option is a version-specific tool for importing a previously exported configuration. PingAccess checks the exported JSON file to ensure that the file is not from a later version of PingAccess and is compatible with application programming interface (API) v3 (PingAccess 5.0 or later).

> **Important**
>
> This operation is destructive and overwrites any existing PingAccess configuration.

> **Note**
>
> Large PingAccess configurations can take several hours to import. During an import, you cannot modify or read the PingAccess configuration.

To restore your PingAccess configuration using a JSON file:

Steps

1. If you are restoring on a new environment, copy the saved `pa.jwk` file to the `<PA_Home>/conf/` directory.
2. Click **Settings**, then go to **System > Configuration Export/Import**.
3. Click **Import Configuration** and select the local file that you want to use.
4. Click **Import**.

5. Click **Confirm** and make any changes indicated by the import process.

   Troubleshooting:

   If the import fails, click **View failures from last import** to view all of the errors logged during the import.

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 replica administrative node and engine nodes are using the proper engine keys, and if they aren't, re-save them to generate a new public key, and reconfigure them to use the newly generated key.

   For more information, see Clustering in PingAccess.

Upgrading PingAccess

This section provides instruction on how to upgrade PingAccess to the latest version depending on what type of environment you have.

After reviewing the Upgrade considerations, review the instructions specific to your environment:

- Upgrading a PingAccess standalone version using the upgrade utility
- Upgrading a PingAccess cluster using the upgrade utility
- Upgrading PingAccess using the Windows installer
- Upgrading a PingAccess standalone version using the incremental update package
- Upgrading a PingAccess cluster using the incremental update package

See PingAccess zero downtime upgrade if you have a clustered environment and want to upgrade PingAccess without impacting resource availability.

For more information on what to do during or after your upgrade, see:

- Performing post-upgrade tasks
- Restoring a PingAccess configuration backup
- Upgrade Troubleshooting
- Upgrade utility configuration file reference

Upgrade considerations

Specific changes in PingAccess might require additional steps during an upgrade to the latest version.

Java 11

Ping Identity intends to remove Java 11 support from PingAccess in December 2025. You can find more information, including Java 17 support, in Installation requirements.

Performing a zero downtime upgrade from 6.0, 6.0.1, 6.0.2, or 6.0.3 to a later version

If you're using PingAccess 6.0, 6.0.1, 6.0.2, or 6.0.3, zero-downtime upgrades to later versions might fail because of PKCE changes.

To prevent this issue, edit your existing web sessions and enable PKCE support:

1. Click **Access** and then go to **Web Sessions > Web Sessions**.
2. Expand the web session and click the **Pencil** icon.
3. Click **Show Advanced**.
4. Click **Enable PKCE**.
5. Edit the web session. Click **Save**.

6. Repeat these steps for each web session.

New templates for error and logout pages

PingAccess 6.1 updated several error and logout page template files to modernize their appearance and remove Ping branding:

- `general.loggedout.page.template.html`
- `general.error.page.template.html`
- `admin.error.page.template.html`
- `policy.error.page.template.html`

You can find more information about the templates in User-facing page customization reference. If you have previously customized the template files, you can re-customize them using the new files.

Using a proxied PingFederate deployment

PingAccess 6.2 introduced the ability to configure a proxied PingFederate deployment through PingAccess. If you have manually configured a similar deployment in an earlier version of PingAccess, you can continue to use it.

However, if you plan to switch from a deployment that you configured manually to a proxied PingFederate deployment through PingAccess, review the configuration options in the proxied PingFederate deployment admin console to verify that it can manage the use cases of your current configuration. Remove any PingFederate-related applications before migrating the configuration.

SPA support

The single-page application (SPA) support checkbox was removed from the admin UI for Web type applications in PingAccess 6.2. Consequently, all Web type applications created in the admin UI in PingAccess 6.2 and later have SPA support enabled by default.

The SPA support checkbox is still available for application programming interface (API) type applications and Web + API type applications. You can find more information in Application field descriptions.

> **Note**
>
> If the default settings for SPA support with Web type applications aren't compatible with your environment or with a specific application, change the default authentication policy or create an authentication challenge policy to replace these settings and achieve the desired behavior. Learn more in Changing the default authentication challenge policy and Configuring authentication challenge policies.

PingAccess 7.1 added a system-provided authentication challenge policy that disables SPA support by default. This policy is useful if you aren't onboarding any new SPAs and don't have many SPAs in your current environment. Learn more in PingAccess 7.1 (June 2022) and Authentication.

Runtime state clustering removal

Support for runtime state clustering was removed in PingAccess 7.0. However, one benefit of runtime state clustering was that it enabled rate limiting rules to behave more consistently in a clustered environment. This was because runtime state clustering enabled all of the engines in a cluster to know the total number of requests for a resource, not just the requests which that engine received.

If you're using runtime state clustering with rate limiting rules, before upgrading to PingAccess 7.0 or later, you should either:

- Configure a load balancer sitting in front of a PingAccess cluster to stick the session to a specific engine. This ensures that a single PingAccess engine node applies the rate limiting rule. Learn more in Managing load balancing strategies.
- Tune down the **Max Burst Requests** interval on the rate limiting rule, following the *<current max burst requests interval>/<number of engines in cluster>* ratio.

Upgrading to or past version 7.3 with a customized `log4j2.xml` file

PingAccess 7.3 introduced a new `log4j-categories.xml` file to enable adjustment of the amount of detail included in PingAccess's logs. Learn more in Configuring verbose logging in the admin console.

If you have customized your `log4j2.xml` file, you must merge this file with the `log4j-categories.xml` file the first time that you upgrade to PingAccess 7.3 or a later version.

Elliptic Curve key pair issues with AWS CloudHSM Client SDK 5

As of PingAccess 7.3, PingAccess offers support for Amazon Web Services (AWS) CloudHSM Client SDK 5 instead of Client SDK 3.

Client SDK 5 introduces an issue with elliptic curve (EC) key pairs for all TLS handshakes, similar to the extant issue with TLS 1.3 for EC and RSA keys. As a result, you can create EC key pairs in PingAccess, but you can't assign them to a listener.

Improved configuration replication compatibility

PingAccess 7.3 introduced the ability for engine nodes and the replica administrative node to connect to an administrative node that's running a later version of PingAccess. This ability was backported to PingAccess 7.2.2 as well.

Nodes running PingAccess 7.2.2 or later can replicate data that's relevant for the version of PingAccess that they're running from the administrative node. You can find more information on clustering and configuration data replication in Clustering in PingAccess.

This ability to maintain compatibility reduces the possibility for outages caused by outdated information, providing more flexibility during the upgrade process for those with large scale or hybrid environments. It also improves stability for containerized deployments because clustered engine nodes don't need to maintain their replication data throughout a restart.

> **Important**
>
> You should still finish upgrading the engine nodes as soon as possible and avoid making configuration changes until all engines have been upgraded.

Upgrading to or past version 8.0 from version 6.2 or below

If you have PingAccess 6.2 or below, you cannot upgrade directly to PingAccess 8.0. You must upgrade to a version above 6.2 first, and then upgrade to 8.0.

This is because in PingAccess 8.0, an outdated H2 JAR file was removed, and PingAccess 6.2 and below use an H2 embedded database.

PingAccess 8.3 enforces encryption of the `pa.keystore.pw` property

PingAccess 8.3 and later enforce encryption of the `pa.keystore.pw` property in the `run.properties` file as follows:

- In non-FIPS mode, if you don't obfuscate `pa.keystore.pw`, PingAccess logs a warning during startup.
- If you try to enable FIPS mode without obfuscating `pa.keystore.pw`, PingAccess terminates startup and logs an error message.

Upgrading a PingAccess standalone version using the upgrade utility

Upgrade a standalone PingAccess deployment to a newer version.

Before you begin

- If you are using PingAccess 3.2 or earlier, upgrade to PingAccess 4.3 or 5.3 before upgrading to the current version of PingAccess.
- Create a backup of your existing PingAccess configuration. If the upgrade fails, 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 Software Development Kit (SDK) for Java, Groovy scripts, and the administrative application programming interface (API). For information on these changes and the actions administrators might need to take, review the Upgrade considerations and the PingAccess 5.0 release notes.

- Verify that you have the following:
  - The PingAccess distribution `.zip` file
  - Your new PingAccess license file, if you plan to switch to a new license file
  - Sign on 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.
- Verify that you are using the same account normally used to run PingAccess.

> **Important**
>
> If you have set `security.overridePropertiesFile=false` in `$JAVA_HOME/jre/lib/java.security`, the upgrade utility might fail because 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 4.0 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 `upgrade.bat` or `upgrade.sh` `-p` parameter. This port configuration is only used for the upgrade. The configured 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-screen while the utility is being run. The upgrade uses an exit code of 0 to indicate a successful upgrade and an exit code of 1 to indicate failure.

Important

If you are upgrading from version 4.3 or earlier, and your installation uses custom plugins, they must be rebuilt using the SDK version included in PingAccess 5.0 or later. Run the upgrade utility manually with the new `-i` command-line option to specify a directory containing the custom plugin JAR files and only the custom plugin JAR files. To migrate your custom plugins, see the PingAccess Addon SDK for Java Migration Guide.

Note

During the upgrade, do not make any changes to the running PingAccess environment.

Steps

1. Copy the `.zip` file for the new PingAccess version to the PingAccess host and extract it.
2. Change to the new version's `/upgrade/bin` directory.
3. Run the PingAccess upgrade utility:

   Choose from:

   - On Windows: `upgrade.bat [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`
   - On Linux: `./upgrade.sh [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`

     Example:

     For example: `./upgrade.sh -p 9002 -i MyJARDir pingaccess-5.3`

Next steps

After you complete the upgrade, see Performing post-upgrade tasks.

PingAccess standalone upgrade parameters

The command-line parameters are the same regardless of the platform and are defined in the following table.

Parameter	Value description
-p *<admin_port>*	Optional port to be used by the temporary PingAccess instance run during the upgrade. The default is 9001.

Parameter	Value description
-i <*directory*>	An optional directory containing additional library JAR files, such as plugins and Java Database Connectivity (JDBC) drivers, to be copied into the target installation. Beginning in version 6.0, JAR files are stored in the `<PA HOME>/deploy` folder. During an upgrade from versions earlier than 6.0, third-party JAR files are migrated from the `lib` folder to the `deploy` folder if no directory is specified. During an upgrade from version 6.0 or later, the contents of the `deploy` folder are migrated to the new `<PA HOME>/deploy` folder if no directory is specified.
<*sourcePingAccessRootDir*>	The `PA_HOME` for the source PingAccess version.
-l <*newPingAccessLicense*>	An optional path to the PingAccess license file to use for the target version. If not specified, the existing license is reused.
-j <*jvm_memory_options_file*>	An optional path to a file with Java Virtual Machine (JVM) options to use for the new PingAccess instance during the upgrade.
-s | --silent	Run the upgrade with no user input required. To use this option, specify the source version's credentials using environment variables.

Environment variables

You can specify the username and password for the source version using these environment variables:

- `PA_SOURCE_API_USERNAME` – The username for the source version's Admin API. This should be set to Administrator.
- `PA_SOURCE_API_PASSWORD` – The basic authorization password for the Administrator in the source version's Admin API.

Java virtual machine (JVM) memory options

You can include these options in the JVM memory options file. Memory amounts use `m` or `g` to specify the unit.

- `-Xms<amount>` – Minimum heap size
- `-Xmx<amount>` – Maximum heap size
- `-XX:NewSize=<amount>` – Minimum size for the Young Gen space
- `-XX:MaxNewSize=<amount>` – Maximum size for the Young Gen space
- `-XX:+UseParallelGC` – Specifies that the parallel garbage collector should be used

You can copy the existing `<PA_HOME>/conf/jvm-memory.options` file to create a JVM memory options file for the upgrade.

Example

```
#Sample JVM Memory options file
-Xms512m
-Xmx1g
-XX:NewSize=256m
-XX:MaxNewSize=512m
-XX:+UseParallelGC
```

Upgrading a PingAccess cluster using the upgrade utility

Upgrade a PingAccess cluster to a newer version.

Before you begin

- If you are using PingAccess 3.2 or earlier, upgrade to PingAccess 4.3 or 5.3 before upgrading to the latest version.
- 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 PingAccess 5.0 or later, there are potentially breaking changes to the Software Development Kit (SDK) for Java, Groovy scripts, and the administrative SDK. For information on these changes and the actions administrators might need to take, see the Upgrade considerations and the PingAccess 5.0 release notes.

- Verify the following:
  - Each node is using the same PingAccess version. You can check the version by viewing the `<PA_HOME>/lib/pingaccess-admin-ui-<version number>.jar` file.
  - The PingAccess administrative node is running.
  - Basic authentication is configured and enabled for the running PingAccess administrative node.
  - You have the `.zip` bundle for the target version of PingAccess.
- Verify that you are using the same account normally used to run PingAccess.

About this task

Use the PingAccess upgrade utility to upgrade a cluster from PingAccess 4.0 or later, the source version, to the most recent version, the target version.

> **Note**
>
> The upgrade procedure causes some downtime. To upgrade a cluster with no downtime, see the Zero Downtime Upgrade guide.

The upgrade utility starts an instance of PingAccess with an administrative listener on port 9001. You can change this port number using the `upgrade.bat` or `upgrade.sh` `-p` parameter. This port configuration is only used for the upgrade. The configured 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-screen while the utility is being run. The upgrade uses an exit code of 0 to indicate a successful upgrade and an exit code of 1 to indicate failure.

Important

If you are upgrading from version 4.3 or earlier, and your installation uses custom plugins, they must be rebuilt using the SDK version included in PingAccess 5.0 or later. Run the upgrade utility manually with the new `-i` command-line option to specify a directory containing the custom plugin JAR files and only the custom plugin JAR files. To migrate your custom plugins, see the PingAccess Addon SDK for Java Migration Guide.

Note

During the upgrade, do not make any changes to the running PingAccess environment.

Steps

1. On the administrative node, extract the `.zip` file for the target version of PingAccess.
2. Go to the new version's `/upgrade/bin` directory.
3. Run the PingAccess upgrade utility:

   Choose from:

   - On Windows: `upgrade.bat [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`
   - On Linux: `./upgrade.sh [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`
4. Review the upgrade log. If it records any manual post-upgrade tasks:
   1. Stop the source administrative console.
   2. Start the target administrative console using the `<PA_HOME>/bin/run.sh` command on Linux systems or the `<PA_HOME>\bin\run.bat` command on Windows systems.
   3. Perform any manual post-upgrade tasks recorded in the upgrade log.
   4. Shut down the upgraded administrative console.
5. Run the upgrade utility on the replica administrative node.

   Choose from:

   - On Windows: `upgrade.bat [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`
   - On Linux: `./upgrade.sh [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`
6. Run the upgrade utility on each engine node.

   Choose from:

   - On Windows: `upgrade.bat [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`

- On Linux: `./upgrade.sh [-p <admin_port>] [-i <directory>] [-j <jvm_memory_options_file>] [-l <newPingAccessLicense>] [-s | --silent] <sourcePingAccessRootDir>`

7. Shut down the entire cluster.
8. Start the upgraded administrative node.
9. Start the upgraded replica administrative node.
10. Start each upgraded engine node.

Next steps

After you complete the upgrade, see Performing post-upgrade tasks.

PingAccess cluster upgrade parameters

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

Parameter definitions

Parameter	Value description
-r | --disable-config-replication	Disables configuration replication on the admin node. For more information about using this parameter in an upgrade, see the Zero Downtime Upgrade.
-p *<admin_port>*	Optional port to be used by the temporary PingAccess instance run during the upgrade. The default is 9001.
-i *<directory>*	An optional directory containing additional library JAR files, such as plugins, Java Database Connectivity (JDBC) drivers to be copied into the target installation. Beginning in version 6.0, `JAR` files are stored in the `<PA HOME>/deploy` folder. During an upgrade from versions earlier than 6.0, third-party `JAR` files are migrated from the `lib` folder to the `deploy` folder if no directory is specified. During an upgrade from version 6.0 or later, the contents of the `deploy` folder are migrated to the new `<PA HOME>/deploy` folder if no directory is specified.
<sourcePingAccessRootDir>	The PA_HOME for the source PingAccess version.
-l *<newPingAccessLicense>*	An optional path to the PingAccess license file to use for the target version. If not specified, the existing license is reused.
-j *<jvm_memory_options_file>*	An optional path to a file with Java Virtual Machine (JVM) memory options to use for the new PingAccess instance during the upgrade.

Parameter	Value description
-s | --silent	Run the upgrade with no user input required. To use this option, specify the source version's credentials using environment variables.

Environment variables

You can specify the username and password for the source version using these environment variables:

- `PA_SOURCE_API_USERNAME` – The username for the source version's Admin API. This should be set to Administrator.
- `PA_SOURCE_API_PASSWORD` – The basic authorization password for the Administrator in the source version's Admin API.

Java virtual machine (JVM) memory options

These options can be included in the JVM memory options file. Memory amounts use `m` or `g` to specify the unit.

- `-Xms<amount>` – Minimum heap size.
- `-Xmx<amount>` – Maximum heap size.
- `-XX:NewSize=<amount>` – Minimum size for the Young Gen space.
- `-XX:MaxNewSize=<amount>` – Maximum size for the Young Gen space.
- `-XX:+UseParallelGC` – Specifies that the parallel garbage collector should be used.

You can copy the existing `<PA_HOME>/conf/jvm-memory.options` file to create a JVM memory options file for the upgrade.

Example

```
#Sample JVM Memory options file
-Xms512m
-Xmx1g
-XX:NewSize=256m
-XX:MaxNewSize=512m
-XX:+UseParallelGC
```

Upgrading PingAccess using the Windows installer

Upgrade PingAccess if you installed PingAccess using the Windows installer.

Before you begin

- If you are using PingAccess 3.2 or earlier, you must upgrade to PingAccess 4.3 or 5.3 before upgrading to PingAccess 6.0.
- Review the Upgrade considerations.

About this task

> **Important**
>
> If additional JAR files, such as custom plugins and Java database connectivity (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.

   Result:

   The existing installation is detected.
3. To upgrade the installation, click **Yes**.
4. If you are switching to a new license, select a license file and specify a temporary admin port.

   > **Note**
   >
   > The temporary admin port is not required when upgrading a cluster node.
5. Click **Next**.
6. Specify the administrator credentials. Click **Next**.

   > **Note**
   >
   > Administrator credentials are not required when upgrading a cluster node.
7. Click **Finish**.

Next steps

After completing the upgrade, Performing post-upgrade tasks.

Upgrading a PingAccess standalone version using the incremental update package

Upgrade a standalone PingAccess deployment to a newer version using the incremental update package.

Before you begin

- Make a backup copy of the PingAccess home directory. If the upgrade fails, use the backup copy to restore PingAccess.
- Review the release notes for every version between your current version and the target version.
- Verify that you have the following:
  - The PingAccess incremental update `.zip` file for the target version
  - 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.

About this task

Use the PingAccess incremental update bundle to upgrade from PingAccess 6.3 or later, the source version, to the most recent maintenance release for that version of PingAccess, the target version. For example, upgrade PingAccess 6.3 to the most recent maintenance release for 6.3.

Steps

1. Stop PingAccess.
2. Open the `readme` file included in the extracted `.zip` bundle.
3. Make the file changes specified in the `readme` file.
4. Restart PingAccess.

Next steps

After you complete the upgrade, see Performing post-upgrade tasks.

Upgrading a PingAccess cluster using the incremental update package

Upgrade a PingAccess cluster to a newer version using the incremental update package.

Before you begin

- Make a backup copy of the PingAccess home directory. If the upgrade fails, use the backup copy to restore PingAccess.
- Review the release notes for every version between your current version and the target version.
- Verify that each node is using the same PingAccess version. You can check the version by viewing the `<PA_HOME>/lib/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.
- Download the PingAccess incremental update `.zip` file for the target version.

About this task

Use the PingAccess incremental update bundle to upgrade a cluster from PingAccess 6.3 or later, the source version, to the most recent maintenance release for that version of PingAccess, the target version. For example, upgrade PingAccess 6.3 to the most recent maintenance release for 6.3.

> **Note**
>
> This upgrade procedure causes some downtime. To upgrade a cluster with no downtime, see the Zero Downtime Upgrade guide.

Steps

1. Upgrade the administrative node.
    1. Extract the `.zip` file for the target version of PingAccess.
    2. Open the `readme` file included in the extracted `.zip` bundle.
    3. Make the file changes specified in the `readme` file.
2. Upgrade the replica administrative node.
    1. Extract the `.zip` file for the target version of PingAccess.
    2. Open the `readme` file included in the extracted `.zip` bundle.
    3. Make the file changes specified in the `readme` file.
3. Upgrade each engine node.
    1. Extract the `.zip` file for the target version of PingAccess.
    2. Open the `readme` file included in the extracted `.zip` bundle.
    3. Make the file changes specified in the `readme` file.
4. Shut down the entire cluster.
5. Start the upgraded administrative node.
6. Start the upgraded replica administrative node.
7. Start each upgraded engine node.

Next steps

After you complete the upgrade, see Performing post-upgrade tasks.

Performing post-upgrade tasks

After upgrading 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`.

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. Review the HTTP requests configuration to ensure the use of the IP source settings is appropriate for the environment.
3. Stop the source version of PingAccess.
4. Start the target version of PingAccess.

Troubleshooting

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 through 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 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.`	Step 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 PingAccess cookie and OAuth token in requests and PingAccess would apply policy to the requests as configured. In 3.1, however, an application programming interface (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 Uniform Resource Locator (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 might 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 might 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 might 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 might be required.`	Indicates that an availability profile was created for the site during the upgrade. You might 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 might 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 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. Set this to <PropertyName>=<PropertyValue>.`	New security headers properties values are not set during an upgrade to preserve the behavior from the source release in the upgrade. If there is no reason not to in your environment, update the `run.properties` file 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. 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. Update the configuration with the new default value if possible.
`The property <PropertyName> was set to a blank value to maintain compatibility. Set this 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 to maintain backwards compatibility, but the recommended value should be used if possible.
`The host for virtual host <VirtualHost>:<Port> already has a keypair associated with it. The keypair previously associated with this virtual host 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 displays 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 might 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.
`Virtual host <VirtualHost> was not migrated. An existing virtual host 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 might 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')` A script should be constructed that performs additional validation 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 Java Virtual Machine (JVM) or Java Development Kit (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 might appear 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 rule set containing multiple cross-origin resource sharing (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 rule set. 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.
	See 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 appears if new language properties are added between the source and target PingAccess 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 appears if the language properties have changed between the source and target PingAccess 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 appears if the language properties have been removed between the source and target PingAccess 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 appears 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 appears on upgrade to release 4.3 or later 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 appears on upgrade to release 5.1 or later to support the existing logging behavior of application resource IDs as integers. The default behavior of release 5.1 and later 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 the `run.properties` file. This change might require a modification to the audit logging database schema.
`Invalid resource method <Method> was removed from resource <ResourceName> on application <ApplicationName>.`	This message appears 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 appears 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.
`As of PingAccess 6.0, runtime state clustering using JGroups has been deprecated. Deployments relying on runtime state clustering will continue to function but the functionality will be replaced in a future version.`	This message appears on an upgrade to release 6.0 or later. The runtime state clustering feature has been deprecated.

Restoring a PingAccess configuration backup

If an upgrade fails, 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. Extract the backup file to `<PA_HOME>`.
3. Restart PingAccess.

   Result:

   Your PingAccess configuration is reverted to the state in the backup archive that was restored.

Upgrade Troubleshooting

This table lists some potential problems and resolutions you might encounter while upgrading PingAccess.

Issue	Resolution
Upgrade from version 4.3 or earlier fails due to Groovy rule changes.	To verify your Groovy scripts are prepared for the upgrade, review the Groovy development reference guide and the Upgrade considerations.
Custom plugins are missing after upgrade.	Manually add the custom plugins to the `<PA Home>/deploy` directory.

Upgrade utility configuration file reference

This configuration file reference provides an overview of configurable parameters used by the upgrade utility. These parameters are configured in the `<UU_HOME>/conf/run.properties` file.

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 Java Virtual Machine (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

A zero downtime upgrade allows you to upgrade your clustered PingAccess environment to the latest version 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 later, there are minor variations depending on your PingAccess source version. Those variations are clearly described where applicable.

> **Note**
>
> Some steps, particularly those related to working with a load balancer, are dependent on your environment. It is expected that you are familiar with the tasks required by these steps. This document does not offer detailed instruction on performing these tasks.

You can upgrade from any version using the upgrade utility, or you can upgrade from version 6.1 to the latest maintenance release using the incremental update bundle. This procedure includes the steps for both methods.

> **Important**
>
> To achieve a successful upgrade, perform the tasks in this document in the order that they are presented. Deviation from these tasks might result in a failed upgrade, system downtime, or both.

> **Note**
>
> If you are using PingAccess 3.2 or earlier, you must upgrade to PingAccess 4.3 or 5.3 before upgrading to PingAccess 6.1.

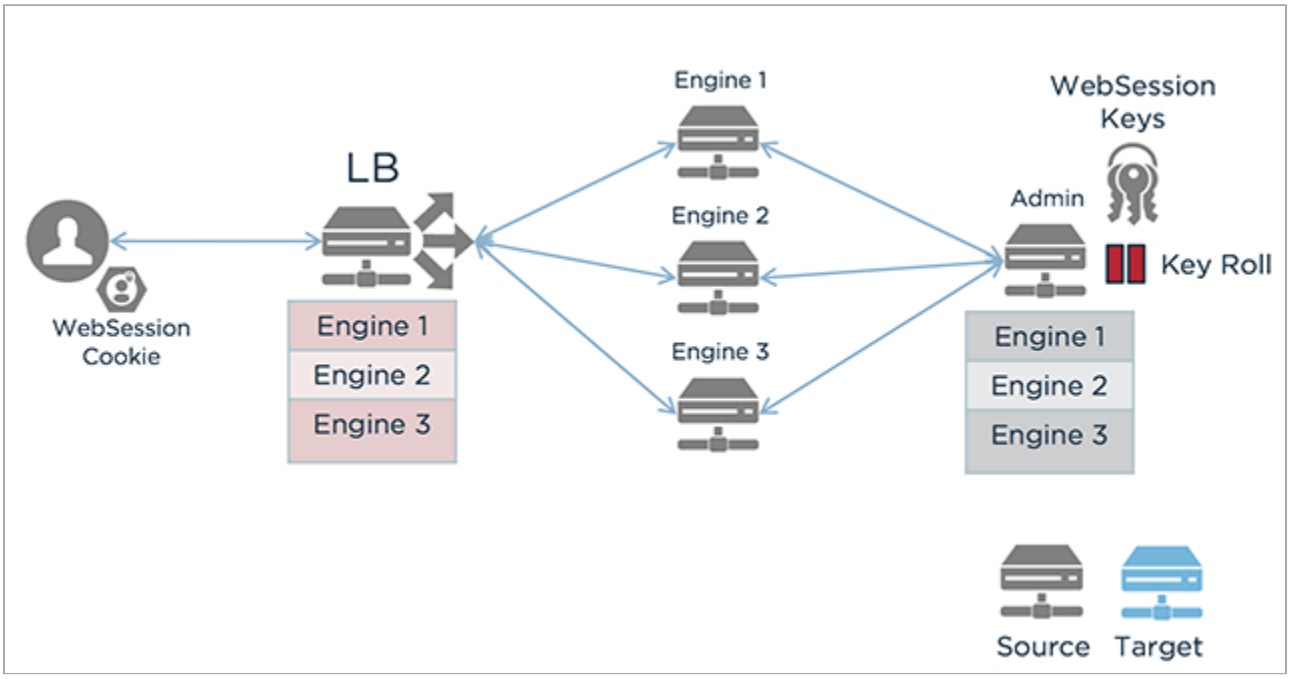
Before you begin, review the Upgrade considerations.

To begin the upgrade process, disable key rolling to prevent active sessions from being invalidated. For more information, see Disabling key rolling.

Disabling key rolling

Disable key rolling to prevent active sessions from being invalidated during the upgrade process. This is a temporary modification, and you will reenable key rolling at the end of the upgrade process.

There are different procedures at this stage depending on the source version of PingAccess. The following flowchart gives an example.



In this flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of three engine nodes, which are all still using the source version of PingAccess.
3. At the Admin node, key rolling has been paused.

Next steps

- Disabling key rolling in PingAccess 6.0 or later
- Disabling key rolling in PingAccess 5.2 or 5.3
- Disabling key rolling in PingAccess 5.0 or 5.1
- Disabling key rolling in PingAccess 4.3 or earlier

Next, you will upgrade the Admin node.

Disabling key rolling in PingAccess 6.0 or later

If the source is PingAccess 6.0 or later, you can disable key rolling.

Steps

1. Click **Access**, then go to **Identity Mappings > Auth Token Management**.
2. In the **Auth Token Management** section, deselect **Key Roll Enabled**.
3. Click **Save**.

4. Click **Access**, then go to **Web Sessions** > **Web Session Management**.
5. In the **Web Session Management** section, deselect **Key Roll Enabled**.
6. Click **Save**.
7. Click **Access**, then go to **Token Validation** > **OAuth Key Management**.
8. In the **OAuth Key Management** section, deselect **Key Roll Enabled**.
9. Click **Save**.

Next steps

Next, you will upgrade the Admin node.

Disabling key rolling in PingAccess 5.2 or 5.3

If the source is PingAccess 5.2 or 5.3, you can disable key rolling.

Steps

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

Next steps

Next, you will upgrade the Admin node.

Disabling key rolling in PingAccess 5.0 or 5.1

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

Steps

1. Go to **Settings → Access → Identity Mappings**.
2. In the **Auth Token Management** section, deselect **Key Roll Enabled**.
3. Click **Save**.

4. Go to **Settings → Access → Web Sessions**.
5. In the **Web Session Management** section, deselect **Key Roll Enabled**.
6. Click **Save**.

Next steps

Next, you will upgrade the Admin node.

Disabling key rolling in PingAccess 4.3 or earlier

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.

Steps

1. Go 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. Go 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 steps

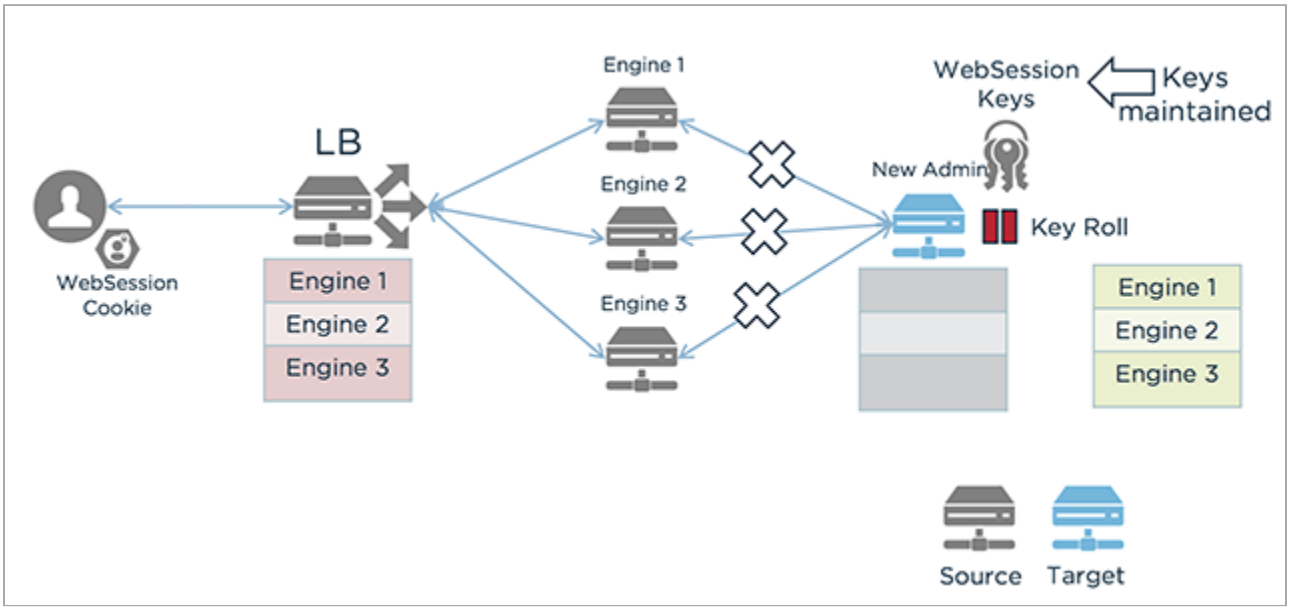
Next, you will upgrade the Admin node.

Upgrading the administrative node

Upgrade the PingAccess administrative node using the PingAccess Upgrade Utility. You will use the `-r` switch to disable configuration replication on the target version.

Before you begin

For more information on upgrading PingAccess, see Upgrading PingAccess.



In the previous flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of three engine nodes, which are all using the source version of PingAccess
3. The administrative node is using the target version of PingAccess.

Before beginning the upgrade process, make sure you have:

- Ensured PingAccess is running
- Downloaded the PingAccess distribution `.zip` file or the incremental update bundle and extracted it.
- The PingAccess license, if you are switching to a new license file
- Administrator credentials
- Basic Authentication enabled

About this task

Any warnings or errors encountered are recorded in `log/upgrade.log`, as well as on the screen while the utility is running. The upgrade uses an exit code of `0` to indicate a successful upgrade and an exit code of `1` to indicate failure.

> **Important**
>
> If you are upgrading from version 4.3 or earlier, and your installation uses custom plugins, they will need to be rebuilt against the new (5.0) Software Development Kit (SDK). You will then run the upgrade utility manually with the new `-i` command-line option to specify a directory containing the custom plugin jars and only the custom plugin jars. To migrate your custom plugins, see the PingAccess Addon SDK for Java Migration Guide.

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

Steps

1. If you are using the upgrade utility, change to the new version's `/upgrade/bin` directory on the command line. For example:

   ```
   cd /pingaccess-6.1.0/upgrade/bin
   ```

2. If you are using the incremental update bundle, disable configuration replication for the replica administrative node.
   1. In a browser, go to https://<*host*>:<*admin-port*>/pa-admin-api/v3/api-docs/.

      Example:

      https://localhost:9000/pa-admin-api/v3/api-docs/

   2. Expand the **/adminConfig/replicaAdmins** endpoint.
   3. Click the **GET /adminConfig/replicaAdmins** operation.
   4. Click **Try it out!** and note the `id` for the replica admin.
   5. Click the **GET /adminConfig/replicaAdmins/{id}** operation.
   6. Enter the `id` of the replica admin you want to update and click **Try it out!**
   7. Copy the **Response Body**.
   8. Click the **PUT /adminConfig/replicaAdmins/{id}** operation and enter the `id` of the replica admin you want to update.
   9. Paste the **Response Body** you copied and change `"configReplicationEnabled"` to `false`.
   10. Click **Try it out!**

       Result:

       If the operation is successful, you will receive a **Response Code** of **200**.

3. If you are using the incremental update bundle, disable configuration replication for each engine node.
   1. In a browser, go to https://<*host*>:<*admin-port*>/pa-admin-api/v3/api-docs/.

      Example:

      https://localhost:9000/pa-admin-api/v3/api-docs/

   2. 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 `false`.
10. Click **Try it out!**

    Result:

    If the operation is successful, you will receive a **Response Code** of **200**.

4. Upgrade the system:

   Choose from:

   - If you are using the upgrade utility on a Windows system, use this command: `upgrade.bat -r <admin_port>] <directory>] <jvm_memory_options_file>] <newPingAccessLicense>] [-s \| --silent] <sourcePingAccessRootDir>`.

     For example:

     ```
     upgrade.bat -r ../pingaccess-5.3.0
     ```

   - If you are using the upgrade utility on a Linux system, use this command: `./upgrade.sh -r <admin_port>] <directory>] <jvm_memory_options_file>] <newPingAccessLicense>] [-s \| --silent] <sourcePingAccessRootDir>`.

     For example:

     ```
     ./upgrade.sh -r ../pingaccess-5.3.0
     ```

   - If you are using the incremental update package, open the readme file and make the file changes specified in the readme.

     > **Important**
     >
     > The `-r` switch will disable configuration replication on the administrative node. You will re-enable configuration replication for each node as part of the upgrade process.

     Parameter definitions

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

Parameter	Value description
-r | --disable-config-replication	Disables configuration replication on the administrative node.
-p <*admin_port*>	Optional port to be used by the temporary PingAccess instance run during the upgrade. The default is 9001.

Parameter	Value description
-i <*directory*>	An optional directory containing additional library JAR files (for example, plugins, JDBC drivers) to be copied into the target installation. Beginning in version 6.0, JAR files are stored in the `<PA HOME>/deploy` folder. During an upgrade from versions earlier than 6.0, third-party JAR files are migrated from the `lib` folder to the `deploy` folder if no directory is specified. During an upgrade from version 6.0 or later, the contents of the `deploy` folder are migrated to the new `<PA HOME>/deploy` folder if no directory is specified.
<*sourcePingAccessRootDir*>	The PA_HOME for the source PingAccess version.
-l <*newPingAccessLicense*>	An optional path to the PingAccess license file to use for the target version. If not specified, the existing license is reused.
-j <*jvm_memory_options_file*>	An optional path to a file with Java Virtual Machine (JVM) memory options to use for the new PingAccess instance during the upgrade.
-s | --silent	Run the upgrade with no user input required. To use this option, specify the source version's credentials using environment variables.

Environment Variables

You can specify the username and password for the source version using these environment variables:

Environment variable	Description
`PA_SOURCE_API_USERNAME`	The username for the source version's Admin application programming interface (API). This should be set to Administrator.
`PA_SOURCE_API_PASSWORD`	The basic authorization password for the Administrator in the source version's Admin API.

JVM Memory options

These options can be included in the JVM memory options file. Memory amounts use `m` or `g` to specify the unit.

Memory option	Description
`-Xms<amount>`	Minimum heap size.
`-Xmx<amount>`	Maximum heap size.
`-XX:NewSize=<amount>`	Minimum size for the Young Gen space.
`-XX:MaxNewSize=<amount>`	Maximum size for the Young Gen space.
`-XX:+UseParallelGC`	Specifies that the parallel garbage collector should be used.

For example:

```
#Sample JVM Memory options file
-Xms512m
-Xmx1g
-XX:NewSize=256m
-XX:MaxNewSize=512m
-XX:+UseParallelGC
```

You can copy the existing `PA_HOME/conf/jvm-memory.options` file to create a JVM memory options file for the upgrade.

5. Stop the existing PingAccess admin instance.
6. Start the new PingAccess admin instance.

Next steps

> **Important**
>
> If PingAccess is running as a service, and you upgraded using the upgrade utility:
>
> - 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 administrative node, you can upgrade the replica admin node.

Upgrading the replica administrative node

Upgrade the PingAccess replica administrative node using the PingAccess Upgrade Utility, then resume configuration replication.

About this task

Any warnings or errors encountered are recorded in `log/upgrade.log`, as well as on the screen while the utility is running. The upgrade uses an exit code of `0` to indicate a successful upgrade and an exit code of `1` to indicate failure.

> **Note**
>
> During the upgrade, it is important to not make any changes to the running PingAccess environment.

Steps

1. If you are using the upgrade utility, change to the new version's `/upgrade/bin` directory on the command line.

   Example:

   ```
   cd /pingaccess-6.1.0/upgrade/bin
   ```

2. Upgrade the system:

   Choose from:

   - If you are using the upgrade utility on a Windows system, use this command: `upgrade.bat <admin_port>] <directory>] <jvm_memory_options_file>] <newPingAccessLicense>] [-s \| --silent] <sourcePingAccessRootDir>`

     For example:

     ```
     upgrade.bat ../pingaccess-5.3.0
     ```

   - If you are using the upgrade utility on a Linux system, use this command: `./upgrade.sh <admin_port>] <directory>] <jvm_memory_options_file>] <newPingAccessLicense>] [-s \| --silent] <sourcePingAccessRootDir>`

     For example:

     ```
     ./upgrade.sh ../pingaccess-5.3.0
     ```

   - If you are using the incremental update package, open the `ReadMeFirst.txt` file and make the file changes specified in the readme.

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

     Parameter definitions

Parameter	Value description
-p <*admin_port*>	Optional port to be used by the temporary PingAccess instance run during the upgrade. The default is 9001.

Parameter	Value description
-i <*directory*>	An optional directory containing additional library JAR files (for example, plugins, JDBC drivers) to be copied into the target installation. Beginning in version 6.0, JAR files are stored in the `<PA_HOME>/deploy` folder. During an upgrade from versions earlier than 6.0, third-party JAR files are migrated from the `lib` folder to the `deploy` folder if no directory is specified. During an upgrade from version 6.0 or later, the contents of the `deploy` folder are migrated to the new `<PA_HOME>/deploy` folder if no directory is specified.
<*sourcePingAccessRootDir*>	The *PA_HOME* for the source PingAccess version.
-l <*newPingAccessLicense*>	An optional path to the PingAccess license file to use for the target version. If not specified, the existing license is reused.
-j <*jvm_memory_options_file*>	An optional path to a file with Java Virtual Machine (JVM) memory options to use for the new PingAccess instance during the upgrade.
-s | --silent	Run the upgrade with no user input required. To use this option, specify the source version's credentials using environment variables.

Environment Variables

You can specify the username and password for the source version using these environment variables:

Environment variable	Description
`PA_SOURCE_API_USERNAME`	The username for the source version's Admin application programming interface (API). This should be set to Administrator.
`PA_SOURCE_API_PASSWORD`	The basic authorization password for the Administrator in the source version's Admin API.

JVM Memory options

These options can be included in the JVM memory options file. Memory amounts use `m` or `g` to specify the unit.

Memory option	Description
`-Xms<amount>`	Minimum heap size.
`-Xmx<amount>`	Maximum heap size.
`-XX:NewSize=<amount>`	Minimum size for the Young Gen space.
`-XX:MaxNewSize=<amount>`	Maximum size for the Young Gen space.
`-XX:+UseParallelGC`	Specifies that the parallel garbage collector should be used.

For example:

```
#Sample JVM Memory options file
-Xms512m
-Xmx1g
-XX:NewSize=256m
-XX:MaxNewSize=512m
-XX:+UseParallelGC
```

You can copy the existing `<PA_HOME>/conf/jvm-memory.options` file to create a JVM memory options file for the upgrade.

3. Stop the existing PingAccess replica admin instance.
4. Start the new PingAccess replica admin instance.

   You're now ready to resume configuration replication for the replica administrative node.
5. In a browser, go to https://*<host>*:*<admin-port>*/pa-admin-api/v3/api-docs/.

   Example:

   https://localhost:9000/pa-admin-api/v3/api-docs/
6. Expand the **/adminConfig/replicaAdmins** endpoint.
7. Click the **GET /adminConfig/replicaAdmins** operation.
8. Click **Try it out!** and note the `id` for the replica admin.
9. Click the **GET /adminConfig/replicaAdmins/{id}** operation.
10. Enter the id of the replica admin you want to update and click **Try it out!**
11. Copy the Response Body.
12. Click the **PUT /adminConfig/replicaAdmins/{id}** operation and enter the id of the replica admin you want to update.
13. Paste the Response Body you copied and change `"configReplicationEnabled"` to `true`.
14. Click **Try it out!**

Result:

If the operation is successful, you will receive a response code of `200`.

15. Click **Settings**, then go to **Clustering > Administrative Nodes**.
16. Ensure the Replica Administrative Node displayed and reporting on the **Administrative Nodes** tab. A healthy node shows a green status indicator.

Next steps

After you have upgraded the administrative and replica administrative nodes, you can begin upgrading the engines.

Upgrading engines

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

> **Important**
>
> 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
- Resuming configuration replication
- 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.

Removing the engine from the load balancer configuration

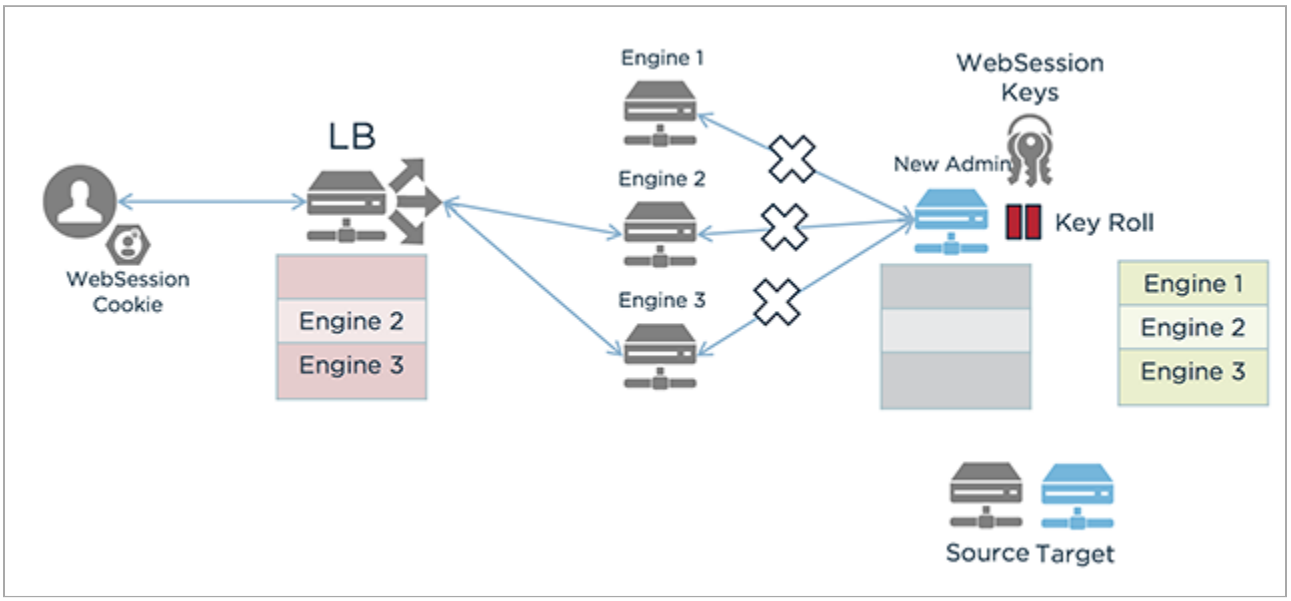
Remove the engine from the load balancer configuration. Because this step is dependent on your environment, no specific instruction will be provided.

Before you begin

You must be 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, 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.

The following flowchart demonstrates engine removal.



In the previous flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of the other two engine nodes. All of the engine nodes are still using the source version of PingAccess.
3. The administrative node is using the target version of PingAccess.

Steps

1. Identify and note the engine you want to upgrade. Ensure you have the engine definition for this engine available.
2. Remove the engine from the load balancer.

   > **Note**
   >
   > Keep a record of the changes you make so that you can reverse this operation later in Adding the engine to the load balancer configuration.

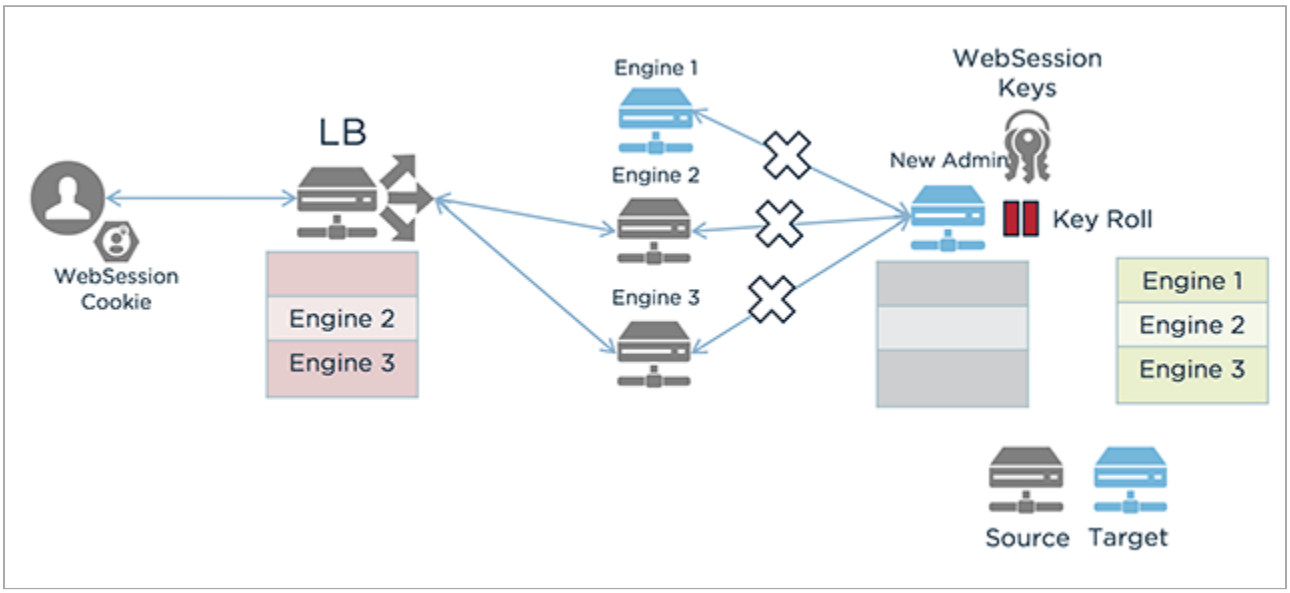
3. Restart the load balancer.

Upgrading the engine

Use the PingAccess Upgrade Utility to upgrade the engine.

Before you begin

For more information on upgrading PingAccess, see Upgrade PingAccess. The following flowchart displays an example engine upgrade.



In this flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of the un-upgraded engine nodes. The first engine node is using the target version of PingAccess, while the other engine nodes are still using the source version of PingAccess.
3. The administrative node is using the target version of PingAccess.

Before beginning the upgrade process, make sure you have:

- Ensured the PingAccess engine is running
- Downloaded the PingAccess distribution `.zip` file or the incremental update bundle and extracted it.
- The PingAccess license

About this task

Any warnings or errors encountered are recorded in `log/upgrade.log`, as well as on the screen while the utility is running. The upgrade uses an exit code of `0` to indicate a successful upgrade and an exit code of `1` to indicate failure.

Steps

1. If you are using the upgrade utility, change to the new version's `/upgrade/bin` directory on the command line.

   Example:

   ```
   cd /pingaccess-6.1.0/upgrade/bin
   ```

2. Upgrade the system:

Choose from:

- If you are using the upgrade utility on a Windows system, use this command: `upgrade.bat <admin_port>] <directory>] <jvm_memory_options_file>] <newPingAccessLicense>] [-s \| --silent] <sourcePingAccessRootDir>`.

  For example:

  ```
  upgrade.bat ../pingaccess-5.3.0
  ```

- If you are using the upgrade utility on a Linux system, use this command: `./upgrade.sh <admin_port>] <directory>] <jvm_memory_options_file>] <newPingAccessLicense>] [-s \| --silent] <sourcePingAccessRootDir>`.

  For example:

  ```
  ./upgrade.sh ../pingaccess-5.3.0
  ```

- If you are using the incremental update package, open the `ReadMeFirst.txt` file and make the file changes specified in the readme.

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

  Parameter definitions

Parameter	Value description
-p <*admin_port*>	Optional port to be used by the temporary PingAccess instance run during the upgrade. The default is `9001`.
-i <*directory*>	An optional directory containing additional library JAR files (for example, plugins, JDBC drivers) to be copied into the target installation. Beginning in version 6.0, JAR files are stored in the `<PA_HOME>/deploy` folder. During an upgrade from versions earlier than 6.0, third-party JAR files are migrated from the `lib` folder to the `deploy` folder if no directory is specified. During an upgrade from version 6.0 or later, the contents of the `deploy` folder are migrated to the new `<PA_HOME>/deploy` folder if no directory is specified.
<*sourcePingAccessRootDir*>	The PA_HOME for the source PingAccess version.
-l <*newPingAccessLicense*>	An optional path to the PingAccess license file to use for the target version. If not specified, the existing license is reused.

Parameter	Value description
-j *<jvm_memory_options_file>*	An optional path to a file with Java Virtual Machine (JVM) memory options to use for the new PingAccess instance during the upgrade.
-s | --silent	Run the upgrade with no user input required. To use this option, specify the source version's credentials using environment variables.

Environment Variables

You can specify the username and password for the source version using these environment variables:

Environment variable	Description
`PA_SOURCE_API_USERNAME`	The username for the source version's Admin application programming interface (API). This should be set to Administrator.
`PA_SOURCE_API_PASSWORD`	The basic authorization password for the Administrator in the source version's Admin API.

JVM Memory options

These options can be included in the JVM memory options file. Memory amounts use `m` or `g` to specify the unit.

Memory option	Description
`-Xms<amount>`	Minimum heap size.
`-Xmx<amount>`	Maximum heap size.
`-XX:NewSize=<amount>`	Minimum size for the Young Gen space.
`-XX:MaxNewSize=<amount>`	Maximum size for the Young Gen space.
`-XX:+UseParallelGC`	Specifies that the parallel garbage collector should be used.

Example:

```
#Sample JVM Memory options file
-Xms512m
-Xmx1g
-XX:NewSize=256m
-XX:MaxNewSize=512m
-XX:+UseParallelGC
```

You can copy the existing `PA_HOME/conf/jvm-memory.options` file to create a JVM memory options file for the upgrade.

3. Stop the existing PingAccess instance. Do not start the new instance.

Next steps

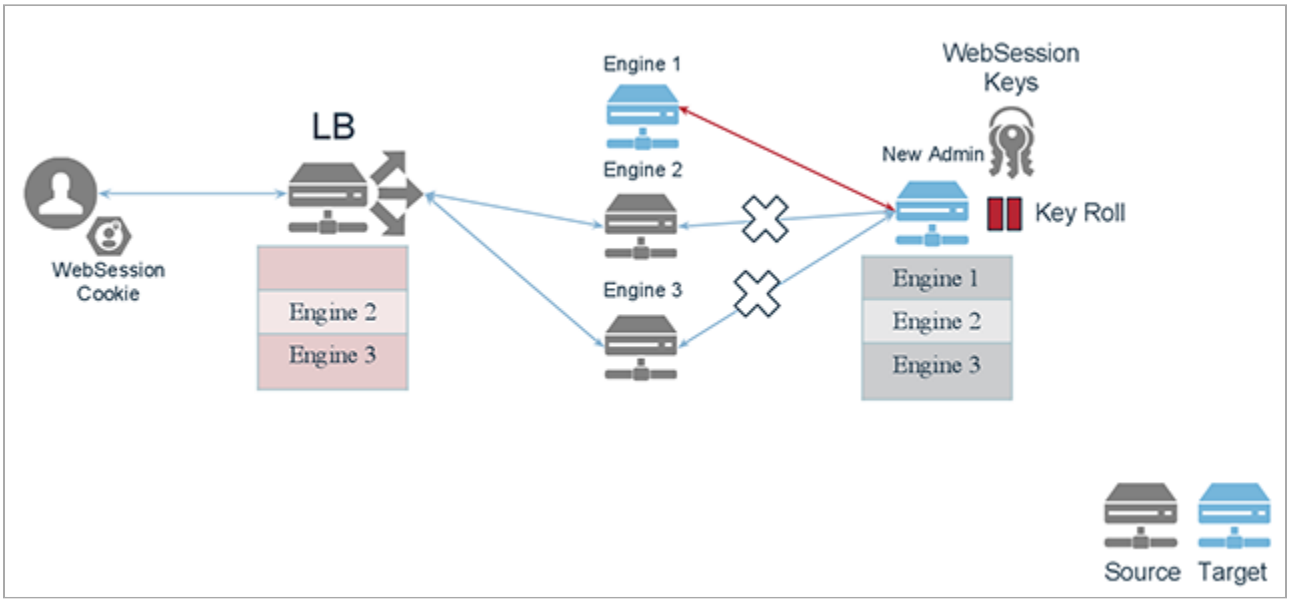
> **Important**
>
> If PingAccess is running as a service and you upgraded using the upgrade utility:
>
> - In Linux, update `PA_HOME` in `/etc/systemd/system/pingaccess.service` to point to the new installation.
> - In Windows, remove the existing PingAccess service (`<OLD_PA_HOME>\sbin\Windows\uninstall-service.bat`) and add the new service (`<NEW_PA_HOME>\sbin\Windows\install-service.bat`).

Resuming configuration replication

Resume the configuration replication that was disabled by the Upgrade Utility. Perform this step for all engine nodes in the cluster.

About this task

You will use the PingAccess Admin API to GET and PUT the relevant configuration data for each of these items.



In the previous flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of the un-upgraded engine nodes. The first engine node is using the target version of PingAccess, and its connection to the administrative node is resuming.
3. The administrative node is using the target version of PingAccess.

> **Note**
>
> Perform the following steps for each engine in the cluster.

To resume configuration replication:

Steps

1. In a browser, go to https://<*host*>:<*admin-port*>/pa-admin-api/v3/api-docs/.

   Example:

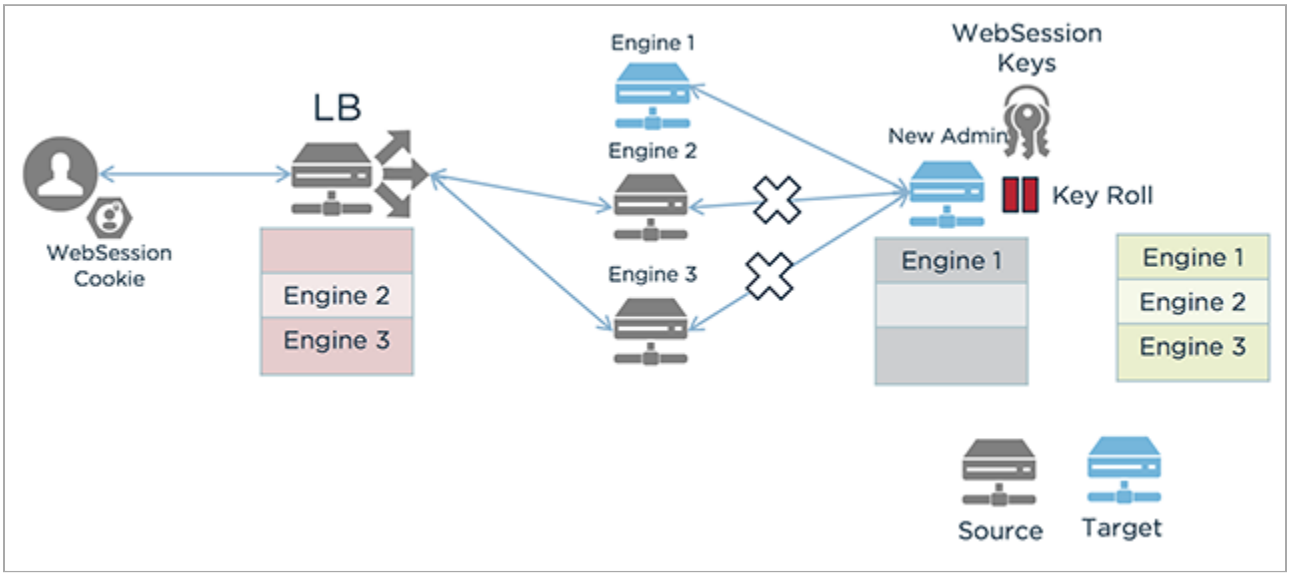
   https://localhost:9000/pa-admin-api/v3/api-docs/

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. In the **ID** field, enter the id of the engine you want to update and click **Try it out!**
7. Copy the entire Response Body.
8. Click the **PUT /engines/{id}** operation and enter the id of the engine you want to update.
9. In the **Engine** field, paste the response body you copied and change `"configReplicationEnabled"` to `true`.
10. Click **Try it out!**

    Result:

    If the operation is successful, you will receive a response code of `200`.
11. Start the node.
12. Repeat the previous steps for each node.
13. Click **Settings**, then go to **Clustering > Engines**.
14. Ensure the engines are displayed and reporting. A healthy engine shows a green status indicator.

    > **Note**
    >
    > There might 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).



In the previous flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of the un-upgraded engine nodes. The first engine node is using the target version of PingAccess, and its connection to the administrative node has resumed.
3. The administrative node is using the target version of PingAccess.

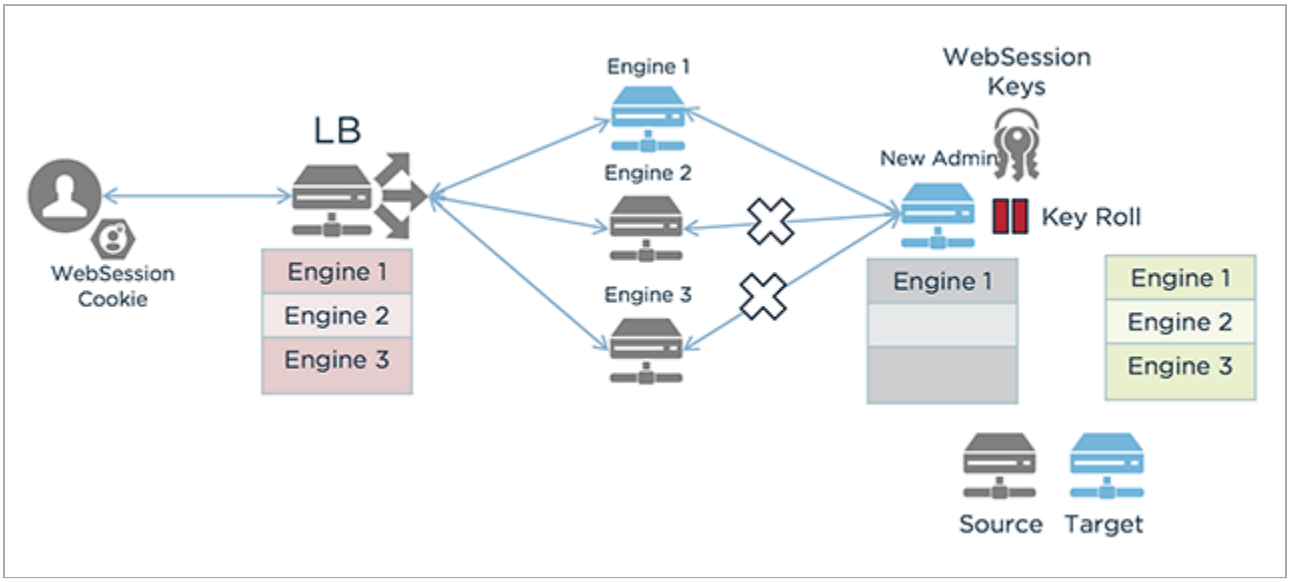
Adding the engine to the load balancer configuration

Add the engine back to the load balancer configuration. Since this step is dependent on your environment, no specific instruction will be provided.

Before you begin

You must be 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.



In the previous flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of the three engine nodes. The first engine node is using the target version of PingAccess, and it has been added to the load balancer configuration.
3. The administrative node is using the target version of PingAccess.

Steps

1. To add the engine to the load balancer configuration, reverse the steps you took in Removing the engine from the load balancer configuration to remove the engine.
2. Restart the load balancer.

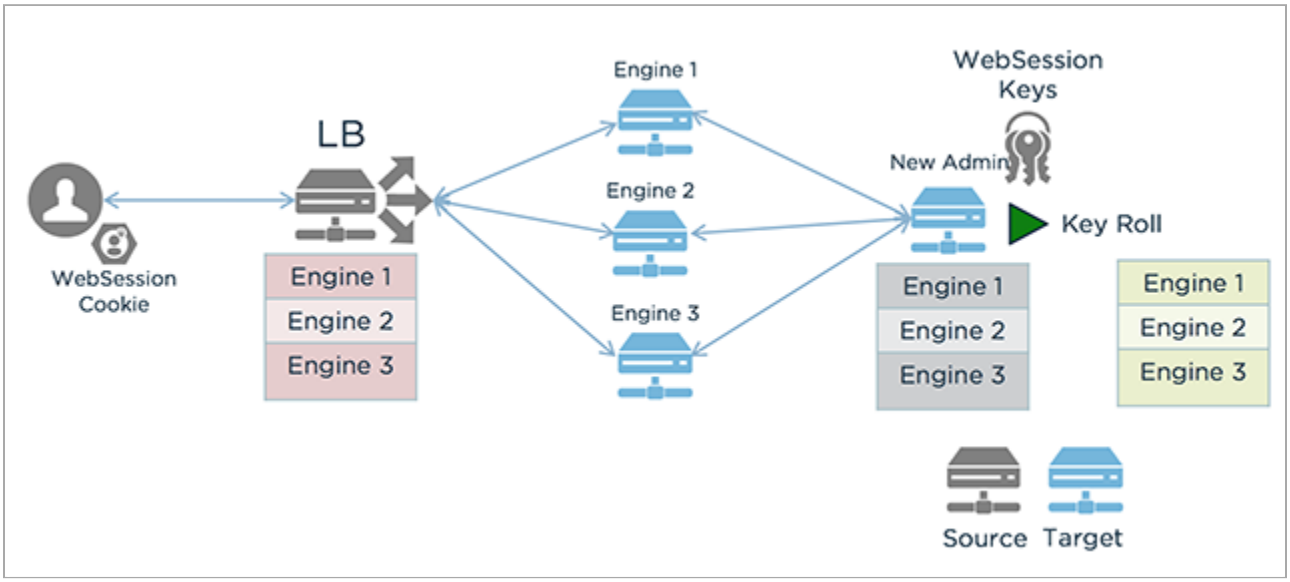
Next steps

Repeat the Upgrading engines 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.

Enabling key rolling

Resume key rolling.

About this task



In the previous flowchart:

1. A user with a WebSession Cookie sends a request to the load balancer.
2. The load balancer directs the request to one of the three upgraded engine nodes. The engine nodes are all using the target version of PingAccess.
3. The administrative node is using the target version of PingAccess.

Steps

1. Click **Access**, then go to **Identity Mappings > Auth Token Management**.
2. In the **Auth Token Management** section, select the **Key Roll Enabled** check box.
3. Verify that the **Key Roll Interval (H)** is correct, then click **Save**.
4. Click **Access**, then go to **Web Sessions > Web Session Management**.
5. In the **Web Session Management** section, select the **Key Roll Enabled** check box.
6. Verify that the **Key Roll Interval (H)** is correct, then click **Save**.
7. Click **Access**, then go to **Token Validation > OAuth Key Management**.
8. In the **OAuth Key Management** section, select the **Key Roll Enabled** check box.
9. Verify that the **Key Roll Interval (H)** is correct, then click **Save**.

Recovering from a failed upgrade

You can recover your PingAccess cluster by switching back to the source version if the upgrade fails.

About this task

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 resume the former installation using these steps.

Steps

1. Stop 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.

Configuring and Customizing PingAccess

This section contains information on how to configure and customize your PingAccess environment.

- For information on configuring PingAccess for server-side session management, see Session management configuration.
- For information on the types of PingAccess logs and how to configure them, see Log configuration.
- For information on customizing PingAccess templates or localizing user-facing documentation and system status messages, see Customize and Localize PingAccess.
- For information on enabling or disabling Federal Information Processing Standards (FIPS) mode, see Managing Federal Information Processing Standards (FIPS) mode.
- If you're using Amazon Web Services (AWS), see Configuring PingAccess to use Amazon Key Management Services.
- For information on formatting environment variables, see Use environment variables to override configuration settings.

Session management configuration

You can configure PingAccess for server-side session management using PingFederate through web session settings.

Web Sessions

Web sessions define the policy for web application session creation, lifetime, timeouts, and their scope. You can configure multiple 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

Several controls exist to scope the PingAccess (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 PingAccess token is sent.

> **Note**
>
> In addition to these controls, you can adjust parameters, such as session timeout, to match the policy requirements of each application.

You must define corresponding OAuth clients 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.

Server-side session management configuration

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

1. PingAccess can reject a PingAccess cookie associated with a PingFederate session that was invalidated because of an end-user driven sign-off.
2. The end user can use a centralized sign-off to sign off from all PingAccess-issued web sessions simultaneously.

The first of these scenarios provides increased scalability and security, ensuring termination of the PingFederate session and rejection of subsequent session validation requests. This scenario implies a user sign-off from resources protected by PingAccess through invalidation of the related PingFederate session. You must make configuration changes in PingAccess to implement this first scenario.

The second scenario provides improved performance and end user experience. When the user explicitly signs off of the PingAccess-issued session, all related PingAccess cookies are deleted, ensuring that the client is no longer authenticated to resources protected by PingAccess. In this scenario, the user has explicitly signed off from all of those protected services. For this second scenario, the user driven sign-off can go directly to the centralized sign-off provider, or PingAccess can initiate the process with the configured token provider.

To learn more, see the following topics:

- Configuring PingFederate for session management
- Configuring PingFederate for user-initiated single logout
- Configuring PingAccess for server-side session management

Configuring PingFederate for session management

Configure PingFederate to revoke PingAccess session cookies.

Steps

1. Sign on to the PingFederate Administrative Console
2. If you are using PingFederate 10.0 or earlier, go to **Server Configuration → Server → Protocol settings → Roles & Protocols** and ensure that **Enable OAuth 2.0 Authorization Server (AS) role** and **OpenID Connect** are enabled.
3. Go to **System → OAuth Settings → Authorization Server Settings** and configure the authorization server settings.
4. Go to the client management section.

   Choose from:

   - If you are using PingFederate 10.0 or earlier, go to **System → OAuth Settings → Client Management**.
   - If you are using PingFederate 10.1 or later, go to **Applications → OAuth → Clients**.

5. Create or modify an existing client.
6. Ensure that **Client Secret** is enabled, and then enter a client secret to be used by PingAccess for authentication.
7. Grant access to the Session Revocation API.

   Choose from:

   - If you are using PingFederate 10.0 or earlier, in the **OpenID Connect** section of the client's configuration page, enable **Grant Access to Session Revocation API**.
   - If you are using PingFederate 10.1 or later, beside **Session API Endpoints**, select **Allow Access to Session Revocation API**.

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

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

Configuring PingFederate for user-initiated single logout

Configure PingFederate to provide PingAccess with access to the PingFederate-managed session.

Steps

1. Sign on to the PingFederate administrative console.
2. Go to **System → OAuth Settings → Authorization Server Settings**.
3. Select **Track User Sessions for Logout**.
4. Click **Save**.
5. Select an OpenID Connect policy.

   Choose from:

   - If you are using PingFederate 10.0 or earlier, go to **System → OAuth Settings → OpenID Connect Policy Management** and click an existing policy.
   - If you are using PingFederate 10.1 or later, go to **Applications → OAuth → OpenID Connect Policy Management** and click an existing policy.

6. On the **Manage Policy** tab, select **Include Session Identifier in ID Token**.

   For more information about configuring an OpenID Connect (OIDC) Policy, see Configuring OpenID Connect Policies in the PingFederate Administrator's Manual.

7. Click **Save**.
8. Select the client to be used by PingAccess.

Choose from:

- If you are using PingFederate 10.0 or earlier, go to **System → OAuth Settings → Client Management** and select the client to be used by PingAccess.
- If you are using PingFederate 10.1 or later, go to **Applications → OAuth → Clients** and select the client to be used by PingAccess.

9. In the **OpenID Connect** section of the client's configuration page, select **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**.

Configuring PingAccess for server-side session management

Configure PingAccess to enable server-side session management.

Steps

1. Sign on to the PingAccess administrative console.
2. Click **Access**, then go to **Web Sessions > Web Sessions**.
3. Click either Create a new web session or Edit an existing web session.
4. Enter a unique **Name** for the web session, up to 64 characters, including special characters and spaces.
5. Specify the **Audience** that the PingAccess token is applicable to, represented as a short, unique identifier between 1 and 32 characters.

   Requests are rejected that contain a PingAccess token with an audience that differs from what is configured in the web session associated with the target application. Changing this setting might affect existing ongoing sessions, forcing the user to re-authenticate to access protected resources.
6. In the **Client ID** field, enter the Client ID defined in PingFederate.
7. In the **Client Credentials Type** section, select **Secret**, and then enter the **Client Secret** associated with the specified Client ID.
8. Click **Show Advanced**.
9. To enable the server-side session management feature, select **Validate Session**.
10. Click **Save**.

Log configuration

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

- You can find information on types of logging in Types of Logging.
  - You can find information on logging in the context of monitoring in Logging, reporting, and troubleshooting.
  - You can find information on logging and auditing in the context of performance tuning in Logging and Auditing.
- You can find information on configuring and managing logging in Configure logging in PingAccess.
  - You can find information on configuring log levels in the administrative console in Log settings also.
- You can find information on logging in the context of troubleshooting specific issues in Log traffic for troubleshooting.
- You can find information on Common Event Format (CEF), JSON format, or logging formats usable by Splunk or a database in Other logging formats.

Types of Logging

Understand the different types of logging that PingAccess offers.

PingAccess logging is handled by a high performance, asynchronous logging framework. You can find more information in Logging. PingAccess provides the following additional types of logging:

Audit logging

Logs a select subset of transaction log information at runtime plus additional details meant to facilitate security auditing and regulatory compliance. If you don't require auditing for interactions with a resource or interactions between PingAccess and PingFederate, it's most efficient to disable audit logging.

> **Important**
>
> If you use audit logging, you should take appropriate steps to secure your audit log files. You can find more information about security measures or audit logging in Security audit logging.

> **Note**
>
> You can also configure PingAccess to write audit log files in Common Event Format (CEF) or JSON format or to write log files to Splunk or a database. Learn more in Other logging formats.

HAR file audit logging

Logs detailed records of specific transactions and sub-transactions between PingAccess and other systems, such as the configured OAuth authorization server (OAuth AS) or a system acting on behalf of the end user.

> **Important**
>
> HAR-formatted audit log files are significantly larger than other log files and can include credentials. You should either carefully configure regex filters to exclude credential information or enable these logs only for troubleshooting purposes. Delete the files when they are no longer necessary.
> You can find more information about HAR file audit logging and regex filters in Log traffic for troubleshooting.

Garbage collection logging

Logs details related to each occurrence of Java garbage collection.

PingAccess logs Java garbage collection data by default, but you can configure garbage collection properties or disable this type of logging. Learn more in Garbage collection logging.

Agent inventory logging

Logs details about your PingAccess agents. Adding the optional header `vnd-pi-agent` to an agent allows it to communicate information about itself and its deployment environment to PingAccess.

Learn more in Agent inventory logging.

Cookie logging

Logs information about the PingAccess cookie, which contains all request identity mappings and the access token from PingFederate, if PingFederate is the AS.

Cookie logging is an optional feature in the `TRACE` log level. It isn't enabled by default. Learn more in Enabling cookie logging.

Logging

PingAccess logging is handled by the log4j2 asynchronous logging library. This library is configurable in the `conf/log4j2.xml` file.

> **Note**
>
> Audit logs are also configurable in `conf/log4j2.xml`. These logs record a select subset of transaction log information at runtime plus additional details. For more information, see Security Audit Logging.

By default, logging information outputs 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.

You can change these defaults by locating and modifying the following properties in the `<Appenders>` section of `conf/log4j2.xml`:

- To rename the log file, modify:

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

- To set the maximum log size, modify:

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

- To set the maximum number of log files, modify:

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

In addition to the standard log4j2 items, PingAccess adds a custom item which is usable in the `log4j2.xml` `<PatternLayout>` configuration. This custom item, `exchangeId`, identifies the ID for a specific request-response pair.

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

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

Note

The `%X` conversion character is required to ensure that the `exchangeID` displays properly.

Security audit logging

PingAccess audit logs record a select subset of transaction log information at runtime plus additional details meant to facilitate security auditing and regulatory compliance.

You can find the audit logs in `<PA_HOME>/log/`. The audit log configuration table describes the elements that the audit logs record. You can configure these elements in `conf/log4j2.xml`.

Important

Because log files can be viewed or modified using a variety of common applications, it's possible for log files to be manipulated to include untrusted or malicious data. You should take appropriate steps to secure your log files. Do not open them in applications that could allow for data execution, such as internet browsers or Microsoft Office products. Instead, open your log 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, Security Token Service (STS), OAuth 2.0, and JSON Web Signature (JWS) requests.

`pingaccess_api_audit.log`

Records PingAccess administrative application programming interface (API) transactions. These transactions represent activity in the PingAccess administrative console. This log also records transaction activity if you're using scripts to configure PingAccess.

`pingaccess_agent_audit.log`

Records transactions between PingAccess agents and the PingAccess engine.

`pingaccess_sideband_client_audit.log`

Records transactions sent to and from the sideband client integration.

`pingaccess_sideband_audit.log`

Records the end-user transactions that the sideband client request captures.

Element	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`	Specifies the pattern type of the path prefix, `Wildcard` or `Regex`.
`AUDIT.authMech`	The 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`	The Internet Protocol (IP) address of the requesting client.

Element	Description
`AUDIT.failedRuleName`	The name of the rule that failed. If there was no rule failure, this field will be blank. **Note** This element is applicable only to the `pingaccess_engine_audit.log`.
`AUDIT.failedRuleType`	The type of rule that failed. If there was no rule failure, this field will be blank. **Note** This element is applicable only to the `pingaccess_engine_audit.log`.
`AUDIT.failedRuleClass`	The Java class of the rule that failed. If there was no rule failure, this field will be blank. **Note** This element is applicable only to the `pingaccess_engine_audit.log`.
`AUDIT.failedRuleSetName`	The name of the containing rule set that failed. If there was no rule failure, this field will be blank. **Note** This element is applicable only to the `pingaccess_engine_audit.log`.
`AUDIT.host`	The PingAccess host name or IP address.
`AUDIT.targetHost`	The backend target that processed the request and generated a response to the PingAccess engine. This variable is unset when PingAccess generated the response directly.
`AUDIT.method`	The HTTP method of the request. For example, `GET`.
`AUDIT.resource`	The name of the resource used to fulfill the request. **Note** This element is applicable only to the `pingaccess_engine_audit.log`.
`AUDIT.responseCode`	The HTTP status code of the response. For example, `200`.

Element	Description
`AUDIT.requestUri`	The request Uniform Resource Identifier (URI) portion of the request. For example, `/foo/bar`.
`AUDIT.subject`	The subject of the transaction.
`AUDIT.trackingId`	The PingFederate tracking ID. You can use this element to help correlate audit information in the PingAccess audit log with information recorded in the PingFederate audit log. This value depends on whether the application type is `Web` or `API`. If the application type is `Web`, the value is formatted as `tid:<Session_Identifier>`. The *<Session_Identifier>* can be used by the Session Revocation API endpoint[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 formatted as `atid:<Hash>`. The *<Hash>* value dervies from the OAuth Access token for the session, and only serves as an identifier; it can't be used for session revocation.
`AUDIT.reqReceivedMillisec`	The time in milliseconds since 1970 that a client request was first received.
`AUDIT.reqSentMillisec`	The time in milliseconds since 1970 that the agent or engine sent a backchannel or proxy request.
`AUDIT.respReceivedMillisec`	The time in milliseconds since 1970 that the agent or engine received a response from a backchannel call or proxy request.
`AUDIT.respSentMillisec`	The 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 that 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 that PingAccess was waiting for another entity to respond to a backchannel call or proxy request.
`AUDIT.siteUnavailableInfo`	If a site is unavailable, this is reason why the last attempted site target is unavailable.

Element	Description
`AUDIT.agentName`	The name of the agent.
`AUDIT.responder`	The component that generated the response. Valid values are `PingAccess`, `PingOne`, `Site`, `Third Party Service`, `OpenID Provider`, and `Authorization Server`. **Note** The `PingOne` responder type refers only to the PingOne Protect integration. When you're using PingOne as a token provider, PingAccess labels backchannel communications as either `OpenID Provider` or `Authorization Server`, depending on the context of the transaction.
`AUDIT.clientCertSerialNum`	The serial number of the client certificate.
`AUDIT.clientCertSubjectDn`	The subject of the client certificate as an X.500 domain name.
`AUDIT.clientCertIssuerDn`	The issuer of the client certificate as an X.500 domain name.
`AUDIT.sidebandName`	The name of the requesting sideband client.
`AUDIT.sidebandDecision`	The policy decision returned in response to the sideband client request. Valid values are `accept` and `reject`.
`agent{a-header-value-key}`	The `vnd-pi-agent` header value for a given key. This represents the header value that an agent sends to PingAccess. Well-known keys are: ***v*** The version of the agent that's making the request. ***h*** The host name of the server where the agent resides. ***t*** The type of agent or the type of platform where the agent resides. This information isn't sent by default. For more information about logging these details, see Agent inventory logging.
`appRequestHeader{a-header-name}`	The HTTP request header value for the given HTTP request header name. Represents the header value that PingAccess sends to the backend site.
`appResponseHeader{a-header-name}`	The HTTP response header value for the given HTTP request header name. Represents the header value that the application sent PingAccess.

Element	Description
`clientRequestHeader{a-header-name}`	The HTTP request header value for the given HTTP request header name. Represents the header value that the client sent PingAccess.
`clientResponseHeader{a-header-name}`	The HTTP response header value for the given HTTP request header name. Represents the header value that PingAccess returned to the client.

Note

To get information about the timing for backchannel calls, such as the OpenID Connect (OIDC) `UserInfo` endpoint call, use the `exchangeID` property to match related log entries and the `AUDIT.roundTripMS` and `AUDIT.proxyroundTripMS` properties to view the timing.

Garbage collection logging

By default, PingAccess logs Java garbage collection data.

The garbage collection log includes details related to each occurrence of garbage collection. For example, the log might record a timestamp and the change in heap memory.

Edit the following properties in the `<PA_HOME>/bin/run.sh` file on Linux systems or the `<PA_HOME>\bin\run.bat` file on Windows systems to configure garbage collection properties.

Property	Description
`GC_FILE=` *"<filename>"*	Specifies the location of the garbage collection log. Comment out this line to disable garbage collection logging.
`GC_FILE_COUNT=` *"<count>"*	Specifies the number of garbage collection files to retain before rotating.
`GC_FILE_SIZE=` *"<size>"*	Specifies the maximum size for garbage collection files.

Agent inventory logging

To log details about your PingAccess agents, you can add custom configuration to the agents and to the PingAccess system.

Agent information isn't included in agent responses by default, except for the agent's name. To include additional information in the logs, customize your PingAccess agents to include the agent header.

Tip

You must edit the `/conf/log4j2.xml` file to log the information included in the agent header. For more information, see Security audit logging.

For information about agent headers, see PAAP agent request.

Agent Header

The optional header `vnd-pi-agent` allows the agent to communicate information about itself and its deployment environment to PingAccess.

The value of this header is a map of comma-separated key-value pairs. An agent can either use the custom keys that are specific to its deployment, or use one or more of the following well-known keys:

v

The version of the agent making the request.

t

The type of agent and/or the type of platform where the agent resides.

h

The hostname of the server where the agent resides.

The syntax for the `vnd-pi-agent` value conforms to a dictionary in this specification, https://datatracker.ietf.org/doc/rfc8941/, where member-values are constrained to be an `sh-string` item.

The following header examples are all considered semantically equivalent:

Example

```
vnd-pi-agent: v="1.0.0", h="apache.example.com", t="Apache 2.4.41"
```

Example

```
vnd-pi-agent: v="1.0.0", h="apache.example.com"
vnd-pi-agent: t="Apache 2.4.41"
```

Example

```
vnd-pi-agent: v="1.0.0"
vnd-pi-agent: h="apache.example.com"
vnd-pi-agent: t="Apache 2.4.41"
```

Configure logging in PingAccess

Common tasks to configure and manage logging in PingAccess.

- For information on how to make your log entries more or less detailed, see Configuring log levels.
- For information on enabling cookie logging, see Enabling cookie logging.
- For information on enabling additional log output destinations, see Appending log messages to syslog and the console.

Configuring log levels

Define log levels for specific package or class names in the `log4j2.xml` file to get more or less detailed logging from a class or group of classes.

About this task

Class or package loggers are defined in the `<AsyncLogger>` `name` attribute. For example, the following line enables cookie logging:

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

> **Note**
>
> If you don't specify a log level for a particular package or class, it inherits the settings for the root logger.

For information on how to configure log levels in the administrative console instead, see Log settings.

To configure the log level for a class or package in the `log4j2.xml` file:

Steps

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

   Example:

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

3. Set the `level` value in the `<AsyncLogger>` element to one of the following values:

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

   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">
   ```

   Change it to:

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

4. Save the modified file.

Result

PingAccess automatically makes the changes effective within 30 seconds.

Enabling cookie logging

Enable cookie logging, which is an optional feature in the `TRACE` log level.

Steps

1. Edit the `conf/log4j2.xml` file 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.

Appending log messages to syslog and the console

Enable additional output destinations, called appenders.

About this task

Console and syslog appenders are pre-configured in `log4j2.xml`, but they're commented out by default.

To enable additional appenders:

Steps

1. Open the `conf/log4j2.xml` file in a text 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's 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 that you want to enable.

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

4. Save the file.

Log traffic for troubleshooting

Enable HTTP Archive (HAR) file audit logging to troubleshoot specific issues or gather more detailed information on the requests and responses sent to and from applications.

When you enable HAR file audit logging for a specific logging category, such as engine audit logging, PingAccess generates a HAR-formatted audit log file for that log category. For example, the HAR-formatted file for the engine audit logging category is the engine audit log HAR file or `<PA_HOME>/log/pingaccess_engine_audit_har.log`.

The HAR-formatted audit logs contain detailed records of specific transactions and sub-transactions between PingAccess and other systems, such as the configured OAuth authorization server (OAuth AS) or a system acting on behalf of the end user.

Audit log records are called transactions. One audit log record equals one transaction. A transaction captures an end-user transaction or an event originating from PingAccess. Sub-transactions are the request-response pairs sent through the HTTP or HTTPS connections.

You can use regex filters to include or exclude certain results from the logs.

> **Important**
>
> HAR format within PingAccess is not exactly equivalent to a browser's HAR format. A standard HAR file is a complete JavaScript Object Notation (JSON) object, whereas the PingAccess HAR-formatted audit log file is a JSON sequence of HAR objects. Each log entry is a complete HAR file, and entries are continually added to the audit log as new information comes in.
> Standard HAR tooling doesn't understand what to do with a JSON sequence, so you must reformat a HAR-formatted audit log before you can view it or parse it with a HAR reader. For more information on how to convert a snapshot instance of the HAR-formatted audit log file into a complete JSON object for parsing, see Parsing HAR-formatted audit log files.

Each HAR file entry within the HAR-formatted audit log file contains an array of audited PingAccess transactions, and each transaction contains one or more request-response pairs. Transactions can provide insight into PingAccess's internal processing.

If an incoming request to PingAccess requires it to make an outgoing request to a site application before it can provide a response, or if you're using an API gateway integration with PingAccess, then a transaction can contain up to two sub-transactions (request-response pairs):

- The first sub-transaction is the original request from the client or API gateway and the response from PingAccess, which it might have modified from the original backend response.
- The second sub-transaction is the request data sent from PingAccess, which PingAccess might have modified from the original request, to the backend site or API gateway and the original response from the site or API gateway.

For most other transactions, such as those seen in agent auditing or requests made from PingAccess to other external systems, there is usually only one sub-transaction.

> **Tip**
>
> PingAccess records entries as request-response pairs in the audit log file, so sub-transactions aren't ordered sequentially. For example:
> Sequentially, a user's inbound request prompts PingAccess to make an outbound request. The response to PingAccess's outbound request then determines PingAccess's final response to the user.
> PingAccess's response to the user derives from the response to its own request, but the audit log records PingAccess's response before the response it receives. This is because the user's request and PingAccess's response to the user pair together as a sub-transaction. Likewise, PingAccess's request pairs with the response it receives.

You can use sub-transactions to:

- Check PingAccess's connections to PingFederate.
- Follow OAuth and OpenID Connect (OIDC) flows to identify where issues might lie.
- Investigate PingAccess policy issues.
- Review the HTTP headers that PingAccess receives to investigate virtual host parsing and base configuration mapping.
- Review the HTTP headers that PingAccess sends.

Metadata

The overview details of the transaction. For a list of specific metadata elements, see Traffic logging reference.

URL

The URL of the transaction. Path varies based on whether the transaction is an input to or output from PingAccess.

HTTP Method

The HTTP method used in the transaction.

HTTP Headers

The HTTP headers included in the transaction. For input to PingAccess, the `Host` and `X-Forwarded-For` headers are common inclusions.

Cookies

The cookies included in the transaction. These can be cookies from PingAccess and PingFederate. If the cookies aren't encrypted, you can view their contents.

> **Note**
>
> The PingAccess cookie contains all of the request's identity mappings and the access token from PingFederate, if PingFederate is the AS. This is all of the information that PingAccess requires to make authorization and mapping decisions.

Responder

The component that generated the response. Valid values are `PingAccess`, `PingOne`, `Site`, `Third Party Service`, `Open ID Provider`, and `Authorization Server`.

> **Note**
>
> The `PingOne` responder type refers only to the PingOne Protect integration. When you're using PingOne as a token provider, PingAccess labels backchannel communications as either `OpenID Provider` or `Authorization Server` depending on the context of the transaction.

Timings

The time that it took to process specific components of the transaction. These components include `%d`, `AUDIT.reqReceivedMillisec`, `AUDIT.reqSentMillisec`, `AUDIT.respReceivedMillisec`, `AUDIT.respSentMillisec`, `AUDIT.roundTripMS`, and `AUDIT.proxyRoundTripMS`. For more information on these components, see Security audit logging.

Failed Policy Components

The rule or rule set name, type, and class of the policy component that failed in the transaction. Failed policy components are only available in the engine log.

PingAccess HAR file responses can contain binary and text from CSS, fonts, JavaScript, and the HTML from PingAccess errors. HAR file responses don't contain HTML responses from the back-end systems, but do indicate the length of the message's HTML body.

> **Tip**
>
> If you require specific HTML from a request or response, enable the `TRACE` log level and check the PingAccess server log to find this information.
> Alternately, you can enable verbose logging for the **HTTP Client** and **HTTP Application** log categories in the PingAccess administrative console. For more information, see Log settings.

API audit log

Requests made to the PingAccess APIs.

Engine audit log

Requests made to or by the PingAccess engine.

Agent audit log

Requests made to or by a PingAccess agent.

Sideband audit log

Requests made to the sideband client.

Sideband client audit log

Requests made to or by the sideband client integration.

For more information on these categories, see Security audit logging.

When you enable HAR file audit logging for a log category, you can filter the types of entries to include in the log and specify what information you want to log. For more information, see the following topics:

- Enabling API audit traffic logging

- Enabling engine traffic logging
- Enabling agent traffic logging
- Enabling sideband traffic logging
- Enabling sideband client traffic logging

> **Important**
>
> HAR-formatted audit log files are significantly larger than other log files and can include credentials. You should either carefully configure regex filters to exclude credential information or enable these logs only for troubleshooting purposes. Delete the files when they are no longer necessary.

Enabling API audit traffic logging

Enable API audit logging including request and responses.

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. In the Logger section, uncomment the `AppenderRef` element for the API audit log HAR file.

   Example:

   ```
   <!-- 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"/>-->
        <AppenderRef ref="ApiAuditLog-HarFile"/>
    </Logger>
   ```

3. In the Appenders section, uncomment the `RollingFile`.

   Example:

```
    <Appenders>
        ...
        <RollingFile name="ApiAuditLog-HarFile"
                             fileName="${sys:pa.home}/log/pingaccess_api_audit_har.log"
                             filePattern="${sys:pa.home}/log/pingaccess_api_audit_har.%d{yyyy-MM-
dd}.log"
                             ignoreExceptions="false">
            <StatusCodeRegExFilter regex=".*"/>
            <HarLogLayout>
                <KeyValuePair key="AUDIT.metadata" value="true"/>
                <KeyValuePair key="AUDIT.http-client" value="true"/>
            </HarLogLayout>
            <Policies>
                <TimeBasedTriggeringPolicy />
            </Policies>
        </RollingFile>
```

4. **Optional:** To filter the entries to add to the log file, edit the value of the `StatusCodeRegExFilter` element.
5. **Optional:** To specify what information to log, add or edit the values in the `HarLogLayout` section of the `RollingFile` element.

   You can add or edit metadata and client response values. For more information, see Traffic logging reference.

Result

Logging begins when the configuration reloads. The configuration reloads at regular intervals according to the `monitorInterval` value.

Enabling engine traffic logging

Enable engine audit logging, including requests and responses.

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. In the Logger section, uncomment the `AppenderRef` element for the engine audit log HAR file.

   Example:

```
       <!-- Audit Log Configuration-->
        ...
        <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"/>-->
            <AppenderRef ref="EngineAuditLog-HarFile"/>
        </Logger>
```

3. In the Appenders section, uncomment the `RollingFile` element for the engine audit log HAR file.

   Example:

```
    <Appenders>
        ...
        <RollingFile name="EngineAuditLog-HarFile"
                     fileName="${sys:pa.home}/log/pingaccess_engine_audit_har.log"
                     filePattern="${sys:pa.home}/log/pingaccess_engine_audit_har.%d{yyyy-MM-
dd}.log"
                     ignoreExceptions="false">
            <StatusCodeRegExFilter regex=".*"/>
            <HarLogLayout>
                <KeyValuePair key="AUDIT.metadata" value="true"/>
                <KeyValuePair key="AUDIT.http-client" value="true"/>
                <KeyValuePair key="AUDIT.http-app" value="true"/>
            </HarLogLayout>
            <Policies>
                <TimeBasedTriggeringPolicy />
            </Policies>
        </RollingFile>
```

4. **Optional:** To filter the entries to add to the log file, edit the value of the `StatusCodeRegExFilter` element.
5. **Optional:** To specify what information to log, add or edit the values in the `HarLogLayout` section of the `RollingFile` element.

   You can add or edit metadata, client response, and app response values. For more information, see Traffic logging reference.

Result

Logging begins when the configuration reloads. The configuration reloads at regular intervals according to the `monitorInterval` value.

Enabling agent traffic logging

Enable agent audit logging, including requests and responses.

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. In the Logger section, uncomment the `AppenderRef` element for the agent audit log HAR file.

   Example:

```
        <!-- Audit Log Configuration-->
         ...
         <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"/>-->
             <AppenderRef ref="AgentAuditLog-HarFile"/>
         </Logger>
```

3. In the Appenders section, uncomment the `RollingFile` element for the engine audit log HAR file.

   Example:

```
    <Appenders>
        ...
        <RollingFile name="AgentAuditLog-HarFile"
                             fileName="${sys:pa.home}/log/pingaccess_agent_audit_har.log"
                             filePattern="${sys:pa.home}/log/pingaccess_agent_audit_har.%d{yyyy-MM-
dd}.log"
                             ignoreExceptions="false">
            <StatusCodeRegExFilter regex=".*"/>
            <HarLogLayout>
                <KeyValuePair key="AUDIT.metadata" value="true"/>
                <KeyValuePair key="AUDIT.http-client" value="true"/>
                <KeyValuePair key="AUDIT.http-app" value="true"/>
            </HarLogLayout>
            <Policies>
                <TimeBasedTriggeringPolicy />
            </Policies>
        </RollingFile>
    </Appenders>
```

4. **Optional:** To filter the entries to add to the log file, edit the value of the `StatusCodeRegExFilter` element.
5. **Optional:** To specify what information to log, add or edit the values in the `HarLogLayout` section of the `RollingFile` element.

   You can add or edit metadata, client response, and app response values. For more information, see Traffic logging reference.

Result

Logging begins when the configuration reloads. The configuration reloads at regular intervals according to the `monitorInterval` value.

Enabling sideband traffic logging

Enable sideband audit logging, including end-user transactions captured by the sideband client request.

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. In the Logger section, uncomment the `AppenderRef` element for the sideband audit log HAR file.

   Example:

   ```
          <!-- Audit Log Configuration-->
           ...
           <Logger name="sidebandaudit" level="INFO" additivity="false">
               <AppenderRef ref="SidebandAuditLog-File"/>
               <!--<AppenderRef ref="SidebandAuditLog-Database-Failover"/>-->
               <!--<AppenderRef ref="SidebandAuditLog-SQLServer-Database-Failover"/>-->
               <!--<AppenderRef ref="SidebandAuditLog-PostgreSQL"/>-->
               <!--<AppenderRef ref="SidebandAudit2Splunk"/>-->
               <AppenderRef ref="SidebandAuditLog-HarFile"/>
           </Logger>
   ```

3. In the Appenders section, uncomment the `RollingFile` element for the engine audit log HAR file.

   Example:

   ```
       <Appenders>
           ...
           <RollingFile name="SidebandAuditLog-HarFile"
                        fileName="${sys:pa.home}/log/pingaccess_sideband_audit_har.log"
                        filePattern="${sys:pa.home}/log/pingaccess_sideband_audit_har.%d{yyyy-MM-
   dd}.log"
                        ignoreExceptions="false">
               <StatusCodeRegExFilter regex="5.."/>
               <HarLogLayout clientBodySizeLimit="16384" appBodySizeLimit="16384">
                   <KeyValuePair key="AUDIT.metadata" value="true"/>
                   <KeyValuePair key="AUDIT.http-client" value="true"/>
                   <KeyValuePair key="AUDIT.http-app" value="true"/>
               </HarLogLayout>
               <Policies>
                   <TimeBasedTriggeringPolicy />
               </Policies>
           </RollingFile>
           ...
       </Appenders>
   ```

4. **Optional:** To filter the entries to add to the log file, edit the value of the `StatusCodeRegExFilter` element.
5. **Optional:** To specify what information to log, add or edit the values in the `HarLogLayout` section of the `RollingFile` element.

   You can add or edit metadata, client response, and app response values. For more information, see Traffic logging reference.

Result

Logging begins when the configuration reloads. The configuration reloads at regular intervals according to the `monitorInterval` value.

Enabling sideband client traffic logging

Enable sideband client audit logging, including transactions sent to or from the sideband client integration.

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. In the Logger section, uncomment the `AppenderRef` element for the sideband client audit log HAR file.

   Example:

   ```
   <!-- Audit Log Configuration-->
    ...
    <Logger name="sidebandclientaudit" level="INFO" additivity="false">
        <AppenderRef ref="SidebandClientAuditLog-File"/>
        <!--<AppenderRef ref="SidebandClientAuditLog-Database-Failover"/>-->
        <!--<AppenderRef ref="SidebandClientAuditLog-SQLServer-Database-Failover"/>-->
        <!--<AppenderRef ref="SidebandClientAuditLog-PostgreSQL"/>-->
        <!--<AppenderRef ref="SidebandClientAudit2Splunk"/>-->
        <AppenderRef ref="SidebandClientAuditLog-HarFile"/>
    </Logger>
   ```

3. In the Appenders section, uncomment the `RollingFile` element for the engine audit log HAR file.

   Example:

   ```
   <Appenders>
       ...
       <RollingFile name="SidebandClientAuditLog-HarFile"
                    fileName="${sys:pa.home}/log/pingaccess_sideband_client_audit_har.log"
                    filePattern="${sys:pa.home}/log/pingaccess_sideband_client_audit_har.%d{yyyy-
   MM-dd}.log"
                    ignoreExceptions="false">
           <StatusCodeRegExFilter regex="5.."/>
           <HarLogLayout clientBodySizeLimit="16384" appBodySizeLimit="16384">
               <KeyValuePair key="AUDIT.metadata" value="true"/>
               <KeyValuePair key="AUDIT.http-client" value="true"/>
               <KeyValuePair key="AUDIT.http-app" value="true"/>
           </HarLogLayout>
           <Policies>
               <TimeBasedTriggeringPolicy />
           </Policies>
       </RollingFile>
       ...
   </Appenders>
   ```

4. **Optional:** To filter the entries to add to the log file, edit the value of the `StatusCodeRegExFilter` element.
5. **Optional:** To specify what information to log, add or edit the values in the `HarLogLayout` section of the `RollingFile` element.

You can add or edit metadata, client response, and app response values. For more information, see Traffic logging reference.

Result

Logging begins when the configuration reloads. The configuration reloads at regular intervals according to the `monitorInterval` value.

Traffic logging reference

You can include these metadata, client, and app elements in PingAccess traffic logs.

Element hierarchy

Each section described here has child elements. If there is a disagreement in settings, the most specific setting is used.

For example, if the metadata element is set to false but the exchange ID is set to true, only the exchange ID is logged. If the metadata element is set to true but the exchange ID is set to false, all metadata elements except the exchange ID are logged.

Limitations

The traffic logs have the following limitations:

- If a request or response body is chunked, only the first chunk is logged by traffic logging.
- Request and response bodies are not decoded.

Metadata elements

You can include metadata elements in the API, engine, and audit traffic logs. These elements provide general information about the logged event.

Item	Description
`AUDIT.metadata`	The section setting for all metadata elements.
`AUDIT.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.

Item	Description
`AUDIT.pathPrefixType`	Indicates the pattern type of the path prefix, `Wildcard` or `Regex`.
`AUDIT.authMech`	The 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`	The Internet Protocol (IP) address of the requesting client.
`AUDIT.failedRuleName`	The name of the rule that failed. If there was no rule failure, this field is blank. **Note** This element is applicable only to the engine log.
`AUDIT.failedRuleType`	Type of rule that failed. If there was no rule failure, this field is blank. **Note** This element is applicable only to the engine log.
`AUDIT.failedRuleClass`	The Java class of rule that failed. If there was no rule failure, this field is blank. **Note** This element is applicable only to the engine log.
`AUDIT.failedRuleSetName`	Name of the containing rule set that failed. If there was no rule failure, this field is blank. **Note** This element is applicable only to the engine log.
`AUDIT.host`	The PingAccess host name or IP address.

Item	Description
`AUDIT.targetHost`	The backend target that processed the request and generated a response to the PingAccess engine. This variable is unset when the response is generated by a target host protected by PingAccess.
`AUDIT.resource`	The name of the resource used to fulfill the request. **Note** This element is applicable only to the engine log.
`AUDIT.subject`	The subject of the transaction.
`AUDIT.trackingId`	The PingFederate tracking ID. You can use this element to help correlate audit information in the PingAccess audit log with information recorded in the PingFederate audit log. This value 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 Session Revocation API endpoint 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 can't be used for session revocation.

The following example shows the metadata section with all elements set to true.

Example

```
<!-- AUDIT.metadata is the section setting for the following fields: -->
<!-- AUDIT.exchangeId to AUDIT.trackingId -->
<KeyValuePair key="AUDIT.metadata" value="true"/>
<KeyValuePair key="AUDIT.exchangeId" value="true"/>
<KeyValuePair key="AUDIT.applicationId" value="true"/>
<KeyValuePair key="AUDIT.applicationName" value="true"/>
<KeyValuePair key="AUDIT.resourceId" value="true"/>
<KeyValuePair key="AUDIT.resourceName" value="true"/>
<KeyValuePair key="AUDIT.pathPrefix" value="true"/>
<KeyValuePair key="AUDIT.pathPrefixType" value="true"/>
<KeyValuePair key="AUDIT.authMech" value="true"/>
<KeyValuePair key="AUDIT.client" value="true"/>
<KeyValuePair key="AUDIT.failedRuleName" value="true"/>
<KeyValuePair key="AUDIT.failedRuleType" value="true"/>
<KeyValuePair key="AUDIT.failedRuleClass" value="true"/>
<KeyValuePair key="AUDIT.failedRuleSetName" value="true"/>
<KeyValuePair key="AUDIT.host" value="true"/>
<KeyValuePair key="AUDIT.targetHost" value="true"/>
<KeyValuePair key="AUDIT.resource" value="true"/>
<KeyValuePair key="AUDIT.subject" value="true"/>
<KeyValuePair key="AUDIT.trackingId" value="true"/>
```

HTTP client elements

Client elements provide information about requests made to PingAccess by clients and the response sent back to the client. For example, a user making a call to the PingAccess administrative API is considered client traffic.

> **Note**
>
> You can include client elements in the API, engine, and audit traffic logs.

Item	Description
`AUDIT.http-client`	The section setting for all client elements.
`AUDIT.http-client-started-date-time`	The date and time of the beginning of the request.
`AUDIT.http-client-time`	The total elapsed time of the request and response.
`AUDIT.http-client-request-method`	The method used in the request.
`AUDIT.http-client-request-target`	The portion of the Uniform Resource Locator (URL) after the host and port.
`AUDIT.http-client-request-http-version`	The HTTP version used by the request.
`AUDIT.http-client-request-cookies`	A list of all the cookies in the request. This is the parent element for `AUDIT.http-client-request-cookie-{cookie}`.

Item	Description
`AUDIT.http-client-request-cookie-{cookie}`	Information about the request cookie with the specified name. You can include this element multiple times for different cookie names.
`AUDIT.http-client-request-headers`	A list of all the headers in the request. This is the parent element for `AUDIT.http-client-request-header-{header}`.
`AUDIT.http-client-request-header-{header}`	Information about the request header with the specified name. You can include this element multiple times for different header names.
`AUDIT.http-client-request-query-strings`	A list of all the parameters and values parsed from the request query string. This is the parent element for `AUDIT.http-client-request-query-string-{query}`.
`AUDIT.http-client-request-query-string-{query}`	Information about the request query string with the specified name. You can include this element multiple times for different query string names.
`AUDIT.http-client-request-post-data-mime-type`	The mime type of the posted request data.
`AUDIT.http-client-request-post-data-text`	The posted request data in plain text format.
`AUDIT.http-client-request-headers-size`	The size, in bytes, of the header from the start of the request to the body.
`AUDIT.http-client-request-body-size`	The size, in bytes, of the request body.
`AUDIT.http-client-response-status-code`	The response status code.
`AUDIT.http-client-response-status-text`	The response status description.
`AUDIT.http-client-response-http-version`	The HTTP version used by the response.
`AUDIT.http-client-response-cookies`	A list of all the cookies in the response. This is the parent element for `AUDIT.http-client-response-cookie-{cookie}`.
`AUDIT.http-client-response-cookie-{cookie}`	Information about the response cookie with the specified name. You can include this element multiple times for different cookie names.

Item	Description
`AUDIT.http-client-response-headers`	A list of all the headers in the response. This is the parent element for `AUDIT.http-client-response-header-{header}`.
`AUDIT.http-client-response-header-{header}`	Information about the response header with the specified name. You can include this element multiple times for different header names.
`AUDIT.http-client-response-content-size`	The size, in bytes, of the response content.
`AUDIT.http-client-response-content-mime-type`	The mime type of the response content.
`AUDIT.http-client-response-content-text`	The response body.
`AUDIT.http-client-response-redirect-url`	The redirect target URL from the location response header.
`AUDIT.http-client-response-headers-size`	The size, in bytes, of the header from the start of the response to the body.
`AUDIT.http-client-response-body-size`	The size, in bytes, of the response body.

The following example shows the client section with all elements set to true.

Example

```
<!-- AUDIT.http-client is the section setting for the following fields: -->
<!-- AUDIT.http-client-started-date-time to AUDIT.http-client-response-body-size -->
<KeyValuePair key="AUDIT.http-client" value="true"/>
<KeyValuePair key="AUDIT.http-client-started-date-time" value="true"/>
<KeyValuePair key="AUDIT.http-client-time" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-method" value="true"/>
<!-- Note: "AUDIT.http-client-request-target" is the target part of the url -->
<KeyValuePair key="AUDIT.http-client-request-target" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-http-version" value="true"/>
<!-- Sets the default value for all client request cookies. -->
<!-- This overrides AUDIT.http-client and is overridden by individual cookie values. -->
<KeyValuePair key="AUDIT.http-client-request-cookies" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-cookie-{cookie}" value="true"/>
<!-- Sets the default value for all client request headers. -->
<!-- This overrides AUDIT.http-client and is overridden by individual header values. -->
<KeyValuePair key="AUDIT.http-client-request-headers" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-header-{header}" value="true"/>
<!-- Sets the default value for all client request query strings. -->
<!-- This overrides AUDIT.http-client and is overridden by individual query strings. -->
<KeyValuePair key="AUDIT.http-client-request-query-strings" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-query-string-{query}" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-post-data-mime-type" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-post-data-text" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-headers-size" value="true"/>
<KeyValuePair key="AUDIT.http-client-request-body-size" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-status-code" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-status-text" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-http-version" value="true"/>
<!-- Sets the default value for all client response cookies. -->
<!-- This overrides AUDIT.http-client and is overridden by individual cookie values. -->
<KeyValuePair key="AUDIT.http-client-response-cookies" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-cookie-{cookie}" value="true"/>
<!-- Sets the default value for all client response headers. -->
<!-- This overrides AUDIT.http-client and is overridden by individual header values. -->
<KeyValuePair key="AUDIT.http-client-response-headers" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-header-{header}" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-content-size" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-content-mime-type" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-content-text" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-redirect-url" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-headers-size" value="true"/>
<KeyValuePair key="AUDIT.http-client-response-body-size" value="true"/>
```

HTTP app elements

App elements provide information about requests made by PingAccess to other tools or services such as PingFederate, and the response sent back to PingAccess. For example, PingAccess making a call to a protected resource is considered app traffic.

> **Note**
>
> You can include app elements in the engine and audit traffic logs.

Item	Description
`AUDIT.http-app`	The section setting for all app elements.
`AUDIT.http-app-started-date-time`	The date and time of the beginning of the request.
`AUDIT.http-app-time`	The total elapsed time of the request and response.
`AUDIT.http-app-request-method`	The method used in the request.
`AUDIT.http-app-request-target`	The portion of the URL after the host and port.
`AUDIT.http-app-request-http-version`	The HTTP version used by the request.
`AUDIT.http-app-request-cookies`	A list of all the cookies in the request. This is the parent element for `AUDIT.http-app-request-cookie-{cookie}`.
`AUDIT.http-app-request-cookie-{cookie}`	Information about the request cookie with the specified name. You can include this element multiple times for different cookie names.
`AUDIT.http-app-request-headers`	A list of all the headers in the request. This is the parent element for `AUDIT.http-app-request-header-{header}`.
`AUDIT.http-app-request-header-{header}`	Information about the request header with the specified name. You can include this element multiple times for different header names.
`AUDIT.http-app-request-query-strings`	A list of all the parameters and values parsed from the request query string. This is the parent element for `AUDIT.http-app-request-query-string-{query}`.
`AUDIT.http-app-request-query-string-{query}`	Information about the request query string with the specified name. You can include this element multiple times for different query string names.
`AUDIT.http-app-request-post-data-mime-type`	The mime type of the posted data.
`AUDIT.http-app-request-post-data-text`	The posted data in plain text format.
`AUDIT.http-app-request-headers-size`	The size, in bytes, of the header from the start of the request to the body.
`AUDIT.http-app-request-body-size`	The size, in bytes, of the request body.

Item	Description
`AUDIT.http-app-response-status-code`	The response status code.
`AUDIT.http-app-response-status-text`	The response status description.
`AUDIT.http-app-response-http-version`	The HTTP version used by the response.
`AUDIT.http-app-response-cookies`	A list of all the cookies in the response. This is the parent element for `AUDIT.http-app-response-cookie-{cookie}`.
`AUDIT.http-app-response-cookie-{cookie}`	Information about the response cookie with the specified name. You can include this element multiple times for different cookie names.
`AUDIT.http-app-response-headers`	A list of all the headers in the response. This is the parent element for `AUDIT.http-app-response-header-{header}`.
`AUDIT.http-app-response-header-{header}`	Information about the response header with the specified name. You can include this element multiple times for different header names.
`AUDIT.http-app-response-content-size`	The size, in bytes, of the response content.
`AUDIT.http-app-response-content-mime-type`	The mime type of the response content.
`AUDIT.http-app-response-content-text`	The response body.
`AUDIT.http-app-response-redirect-uri`	The redirect target URL from the location response header.
`AUDIT.http-app-response-headers-size`	The size, in bytes, of the header from the start of the response to the body.
`AUDIT.http-app-response-body-size`	The size, in bytes, of the response body.

The following example shows the app section with all elements set to true.

Example

```
<!-- AUDIT.http-app is the section setting for the following fields: -->
<!-- AUDIT.http-app-started-date-time to AUDIT.http-app-response-body-size -->
<KeyValuePair key="AUDIT.http-app" value="true"/>
<KeyValuePair key="AUDIT.http-app-started-date-time" value="true"/>
<KeyValuePair key="AUDIT.http-app-time" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-method" value="true"/>
<!-- Note: "AUDIT.http-app-request-target" is the target part of the url -->
<KeyValuePair key="AUDIT.http-app-request-target" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-http-version" value="true"/>
<!-- Sets the default value for all app request cookies. -->
<!-- This overrides AUDIT.http-app and is overridden by individual cookie values. -->
<KeyValuePair key="AUDIT.http-app-request-cookies" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-cookie-{cookie}" value="true"/>
<!-- Sets the default value for all app request headers. -->
<!-- This overrides AUDIT.http-app and is overridden by individual header values. -->
<KeyValuePair key="AUDIT.http-app-request-headers" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-header-{header}" value="true"/>
<!-- Sets the default value for all app request query strings. -->
<!-- This overrides AUDIT.http-app and is overridden by individual query strings. -->
<KeyValuePair key="AUDIT.http-app-request-query-strings" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-query-string-{query}" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-post-data-mime-type" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-post-data-text" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-headers-size" value="true"/>
<KeyValuePair key="AUDIT.http-app-request-body-size" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-status-code" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-status-text" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-http-version" value="true"/>
<!-- Sets the default value for all app response cookies. -->
<!-- This overrides AUDIT.http-app and is overridden by individual cookie values. -->
<KeyValuePair key="AUDIT.http-app-response-cookies" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-cookie-{cookie}" value="true"/>
<!-- Sets the default value for all app response headers. -->
<!-- This overrides AUDIT.http-app and is overridden by individual header values. -->
<KeyValuePair key="AUDIT.http-app-response-headers" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-header-{header}" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-content-size" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-content-mime-type" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-content-text" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-redirect-uri" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-headers-size" value="true"/>
<KeyValuePair key="AUDIT.http-app-response-body-size" value="true"/>
```

Parsing HAR-formatted audit log files

Reformat a snapshot instance of a HAR-formatted audit log file so that you can view it or parse it with a HTTP Archive (HAR) reader.

About this task

> **Note**
>
> Before sharing any HAR data with a third-party application, carefully review the third-party application's permissions and sanitize any potentially sensitive information out of the log files.
>
> - For more information on configuring regex filters for your log files, see Log traffic for troubleshooting.
> - For information on other precautions, see Security audit logging.

Steps

1. Download the jq command-line tool from https://stedolan.github.io/jq/download/.

   Select a jq version for the operating system that you deployed your PingAccess environment on.

   For more information on PingAccess operating system requirements, see System requirements.

2. Create a file called `pa-har-merge.jq`.

   Example:

   ```
   {
   log: {
     version: .[0].log.version,
     creator: .[0].log.creator,
     entries: (reduce .[] as $entry ([]; . + ($entry.log.entries | map(. + { _metadata:
   $entry.log._metadata })))
     }
   }
   ```

   For examples of how to parse the PingAccess HAR-formatted log files with `pa-merge-har.jq`, see the following commands. These examples assume that:

   - You've set `PA_HOME` and `PA_HAR_MERGE_HOME` as environment variables that define the base paths to the PingAccess instance and the `pa-merge-har.jq` file respectively.
   - You're attempting to parse the HAR-formatted API audit log file.

     To filter requests based on request URL, run the command:

     ```
     cat $PA_HOME/log/pingaccess_api_audit_har.log | jq -s -f $PA_HAR_MERGE_HOME/pa-har-merge.jq |
     jq '.log.entries = [ .log.entries[] | select(.request.url != "/pa-admin-api/v3/
     adminSessionInfo/checkOnly") ]
     ```

     To output the HAR-formatted log file into a file format that's usable with a standard HAR viewer, run the command:

     ```
     cat $PA_HOME/log/pingaccess_api_audit_har.log | jq -s -f $PA_HAR_MERGE_HOME/pa-har-merge.jq >
     log.har
     ```

> **Note**
>
> View the output `log.har` file with a standard HAR viewer, such as browser dev tools or the HTTP Archive Viewer.

Other logging formats

Configure PingAccess to write the audit logs in additional formats, such as to a database or to Splunk.

- You can find information on configuring PingAccess to write the audit logs to an Oracle or SQL Server database in Writing logs to databases.
- You can find information on configuring PingAccess to write the audit logs into a file format that Splunk can digest easily in Writing audit logs for Splunk.
- You can find information on configuring PingAccess to write the audit logs in Common Event Format (CEF) in Writing audit logs in Common Event Format.
- You can find information on configuring PingAccess to write the audit logs in JSON format in Writing logs in JSON format.

Writing logs to databases

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

About this task

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

Steps

1. Ensure that your database driver JAR file is installed in the `<PA_HOME>/deploy` directory.
2. After installing the driver, restart PingAccess.
3. In the `conf/log4j2.xml` file, uncomment one or more of the preset appender configurations listed in the following table.

Database	Configuration
Oracle	◦ For administrative application programming interface (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. ◦ For sideband client audit logging, uncomment the `<JDBC>` element with the `name="SidebandClientAuditLog-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements. ◦ For sideband end-user audit logging, uncomment the `<JDBC>` element with the `name="SidebandAuditLog-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.

Database	Configuration
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. ◦ For sideband client audit logging, uncomment the `<JDBC>` element with the `name="SidebandClientAuditLog-SQLServer-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements. ◦ For sideband end-user audit logging, uncomment the `<JDBC>` element with the `name="SidebandAuditLog-SQLServer-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.

Database	Configuration
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. ◦ For sideband client audit logging, uncomment the `<JDBC>` element with the `name="SidebandClientAuditLog-PostgreSQL-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements. ◦ For sideband end-user audit logging, uncomment the `<JDBC>` element with the `name="SidebandAuditLog-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 isn't accessible. Use the `retryIntervalSeconds` attribute to specify the number of seconds that must pass before retrying the primary Java database connectivity (JDBC) appender.

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

Note

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`.

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

Example:

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"/>-->
    <!--<AppenderRef ref="ApiAuditLog-HarFile"/>-->
</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"/>-->
    <!--<AppenderRef ref="EngineAuditLog-HarFile"/>-->
</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"/>-->
    <!--<AppenderRef ref="AgentAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandclientaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandClientAuditLog-File"/>
    <AppenderRef ref="SidebandClientAuditLog-Database-Failover"/>
    <!--<AppenderRef ref="SidebandClientAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="SidebandClientAudit2Splunk"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandAuditLog-File"/>
    <AppenderRef ref="SidebandAuditLog-Database-Failover"/>
    <!--<AppenderRef ref="SidebandAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="SidebandAudit2Splunk"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-HarFile"/>-->
</Logger>
```

Example:

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"/>-->
    <!--<AppenderRef ref="ApiAuditLog-HarFile"/>-->
</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"/>-->
    <!--<AppenderRef ref="EngineAuditLog-HarFile"/>-->
</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"/>-->
    <!--<AppenderRef ref="AgentAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandclientaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandClientAuditLog-File"/>
    <!--<AppenderRef ref="SidebandClientAuditLog-Database-Failover"/>-->
    <AppenderRef ref="SidebandClientAuditLog-SQLServer-Database-Failover"/>
    <!--<AppenderRef ref="SidebandClientAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="SidebandClientAudit2Splunk"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandAuditLog-File"/>
    <!--<AppenderRef ref="SidebandAuditLog-Database-Failover"/>-->
    <AppenderRef ref="SidebandAuditLog-SQLServer-Database-Failover"/>
    <!--<AppenderRef ref="SidebandAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="SidebandAudit2Splunk"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-HarFile"/>-->
</Logger>
```

Example:

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"/>-->
    <!--<AppenderRef ref="ApiAuditLog-HarFile"/>-->
</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"/>-->
    <!--<AppenderRef ref="EngineAuditLog-HarFile"/>-->
</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"/>-->
    <!--<AppenderRef ref="AgentAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandclientaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandClientAuditLog-File"/>
    <!--<AppenderRef ref="SidebandClientAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-SQLServer-Database-Failover"/>-->
    <AppenderRef ref="SidebandClientAuditLog-PostgreSQL"/>
    <!--<AppenderRef ref="SidebandClientAudit2Splunk"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandAuditLog-File"/>
    <!--<AppenderRef ref="SidebandAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-SQLServer-Database-Failover"/>-->
    <AppenderRef ref="SidebandAuditLog-PostgreSQL"/>
    <!--<AppenderRef ref="SidebandAudit2Splunk"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-HarFile"/>-->
</Logger>
```

6. 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 the Oracle, PostgreSQL, or MS SQL Server documentation.

> **Important**
>
> For PostgreSQL database scripts, use of the default `public` schema isn't 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 using 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
> ```

Writing audit logs for Splunk

Ping Identity provides a custom Splunk app for PingAccess to process audit logs generated by a PingAccess deployment.

Before you begin

- Go to the Splunk website and download Splunk.
- Install 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 that you can generate reports, graphs, and other data visualization from.

> **Note**
>
> The PingAccess app for Splunk is available separately. It requires enterprise-licensed (or trial) installation of the Splunk software and the Splunk Universal Forwarder, which collects data from the PingAccess Splunk audit logs. The application includes additional documentation on installation and available features.

The PingAccess app for Splunk provides rich system monitoring and reporting, including:

- Current transaction and system reports
- Service reports, such as a daily usage report and IdP and SP reports per connection
- Trend reports, such as weekly and monthly usage reports, and trend analysis

The application uses a specially formatted version of the audit logs. To write these specially formatted logs to the PingAccess log directory, perform the following steps.

> **Note**
>
> The PingAccess app for Splunk was designed to use the default Splunk log pattern configuration. If you have changed the output format of the Splunk rolling files, those changes could impact the functionality of the PingAccess app for Splunk.

Steps

1. Set up your Splunk server.
    1. Enable a receiver to listen for data from the servers hosting PingAccess.

       For more information, see the Splunk documentation.
    2. Install the PingAccess app for Splunk.

       > **Tip**
       >
       > To download the free application from Splunkbase.splunk.com, search for PingAccess.

       For installation instructions, see the Splunk Add-on documentation.
2. Configure PingAccess to output the following available Splunk audit logs:
    - `pingaccess_engine_audit_splunk.log`
    - `pingaccess_api_audit_splunk.log`
    - `pingaccess_agent_audit_splunk.log`

      > **Note**
      >
      > These logs output to `<PA_HOME>/log/` by default.

        1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
        2. In the `Audit Log Configuration` section, edit the `apiaudit`, `engineaudit`, and `agentaudit` logger configurations to uncomment the Splunk `AppenderRef`.

           Example:

```
<!-- ======================= -->
<!-- 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"/>
    <!--<AppenderRef ref="ApiAuditLog-HarFile"/>-->
</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"/>
    <!--<AppenderRef ref="EngineAuditLog-HarFile"/>-->
</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"/>
    <!--<AppenderRef ref="AgentAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandclientaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandClientAuditLog-File"/>
    <!--<AppenderRef ref="SidebandClientAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandClientAuditLog-PostgreSQL"/>-->
     <AppenderRef ref="SidebandClientAudit2Splunk"/>
    <!--<AppenderRef ref="SidebandClientAuditLog-HarFile"/>-->
</Logger>
<Logger name="sidebandaudit" level="INFO" additivity="false">
    <AppenderRef ref="SidebandAuditLog-File"/>
    <!--<AppenderRef ref="SidebandAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="SidebandAuditLog-PostgreSQL"/>-->
     <AppenderRef ref="SidebandAudit2Splunk"/>
    <!--<AppenderRef ref="SidebandAuditLog-HarFile"/>-->
</Logger>
```

3. Uncomment the `RollingFile` appender references for the `ApiAudit2Splunk`, `EngineAudit2Splunk`, and `AgentAudit2Splunk` `RollingFile` elements.

   Example:

   This is the default configuration for the `ApiAudit2Splunk` file:

```
<!--
<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.authMech}" client="%X{AUDIT.client}" method="%X{AUDIT.method}"
requestUri="%X{AUDIT.requestUri}" responseCode="%X{AUDIT.responseCode}"
responder="%X{AUDIT.responder}" engineHostname="%X{AUDIT.host}" %n</pattern>
    </PatternLayout>
    <Policies>
        <TimeBasedTriggeringPolicy />
    </Policies>
</RollingFile>
 -->
```

This is the updated configuration for the `ApiAudit2Splunk` file, with the `RollingFile` uncommented and no other changes:

```
<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.authMech}" client="%X{AUDIT.client}" method="%X{AUDIT.method}"
requestUri="%X{AUDIT.requestUri}" responseCode="%X{AUDIT.responseCode}"
responder="%X{AUDIT.responder}" engineHostname="%X{AUDIT.host}" %n</pattern>
    </PatternLayout>
    <Policies>
        <TimeBasedTriggeringPolicy />
    </Policies>
</RollingFile>
```

3. Set up the Splunk Universal Forwarder.
    1. Download the Splunk Universal Forwarder from Splunk and install it on the PingAccess server.
    2. Configure the Splunk Universal Forwarder to monitor the three Splunk log files (`pingaccess_engine_audit_splunk.log` , `pingaccess_api_audit_splunk.log` , and `pingaccess_agent_audit_splunk.log`) and forward the data to the receiver you configured.

       For detailed installation and configuration instructions, see the Splunk documentation.

Writing audit logs in Common Event Format

You can configure PingAccess to write any of its five audit logs in Common Event Format (CEF).

About this task

To enable CEF:

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. Select a tab to continue.

   Choose from:

   - If you have a server that supports rsyslog, use the CEF syslog appender tab.
   - If your server does not support rsyslog, use the CEF file tab.

CEF file

Enabling the CEF format file
Steps

1. Uncommment the CEF file appender references in the `apiaudit`, `engineaudit`, `agentaudit`, `sidebandclientaudit`, and `sidebandaudit` logger configurations.

   Example:

   In the `Audit log configuration` section of the `log4j2.xml` file, go to the `apiaudit` logger configuration and uncomment the `ApiAuditLogToCEF-FILE` appender reference:

   ```
   <!-- ======================= -->
   <!-- Audit log configuration -->
   <!-- ======================= -->
   <Logger name="apiaudit" level="${sys:pa.log.level.apiaudit:-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"/>-->
      <!--<AppenderRef ref="ApiAuditLog-HarFile"/>-->
       <AppenderRef ref="ApiAuditLogToCEF-File"/>
      <!--<AppenderRef ref="ApiAuditLogToCEF-Syslog-Failover"/>-->
   </Logger>
   ```

   Repeat this with the `EngineAuditLogToCEF-FILE`, `AgentAuditLogToCEF-FILE`, `SidebandClientAuditLogToCEF-FILE`, and `SidebandAuditLogToCEF-FILE` appender references.

2. Uncomment the `RollingFile` preset appender configurations in the `Api Audit log : CEF format file`, `Engine Audit log : CEF format file`, `Agent Audit log : CEF format file`, `SidebandClient Audit log : CEF format file`, and `Sideband Audit log : CEF format file` sections.

   Example:

   In the `Api Audit log : CEF format file` section, uncomment the `ApiAuditLogToCEF-FILE` `RollingFile` preset appender configuration:

   ```
   <RollingFile name="ApiAuditLogToCEF-File"
                fileName="${sys:pa.home}/log/pingaccess_api_audit_cef.log"
                filePattern="${sys:pa.home}/log/pingaccess_api_audit_cef.%d{yyyy-MM-dd}.log" >
      <PatternLayout>
         <pattern>%escape{CEF}{CEF:0|Ping Identity|PingAccess|%X{AUDIT.paVersion}|%X{exchangeId}|
   API_AccessEvent|0|rt=%d{ISO8601} msg=%X{AUDIT.responseCode} duid=%X{AUDIT.subject}
   src=%X{AUDIT.client} requestMethod=%X{AUDIT.method} request=%X{AUDIT.requestUri}
   cs1Label=AuthenticationMechanism cs1=%X{AUDIT.authMech} cs2Label=RoundTripMS
   cs2=%X{AUDIT.roundTripMS} externalId=%X{AUDIT.trackingId} %n}</pattern>
      </PatternLayout>
      <Policies>
   <TimeBasedTriggeringPolicy />
      </Policies>
   </RollingFile>
   ```

Repeat this with the `EngineAuditLogToCEF-FILE`, `AgentAuditLogToCEF-FILE`, `SidebandClientAuditLogToCEF-FILE`, and `SidebandAuditLogToCEF-FILE` appender configurations.

3. Save and close the file.

CEF syslog appender

Enabling the CEF formatted syslog appender

Steps

1. Uncommment the syslog failover appender references in the `apiaudit`, `engineaudit`, `agentaudit`, `sidebandclientaudit` and `sidebandaudit` sections.

   Example:

   In the `Audit log configuration` section of the `log4j2.xml` file, go to the `apiaudit` logger configuration and uncomment the `<AppenderRef ref="ApiAuditLogToCEF-Syslog-Failover"/>` appender reference:

   ```
   <!-- ======================= -->
   <!-- Audit log configuration -->
   <!-- ======================= -->
   <Logger name="apiaudit" level="${sys:pa.log.level.apiaudit:-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"/>-->
      <!--<AppenderRef ref="ApiAuditLog-HarFile"/>-->
      <!--<AppenderRef ref="ApiAuditLogToCEF-File"/>-->
       <AppenderRef ref="ApiAuditLogToCEF-Syslog-Failover"/>
   </Logger>
   ```

   Repeat this with the `<AppenderRef ref="EngineAuditLogToCEF-Syslog-Failover"/>`, `<AppenderRef ref="AgentAuditLogToCEF-Syslog-Failover"/>`, `<AppenderRef ref="SidebandClientAuditLogToCEF-Syslog-Failover"/>`, and `<AppenderRef ref="SidebandAuditLogToCEF-Syslog-Failover"/>` appender references.

2. Uncomment the `Socket` appender configurations in the `Api Audit log : CEF Formatted syslog appender`, `Engine Audit log : CEF Formatted syslog appender`, `Agent Audit log : CEF Formatted syslog appender`, `SidebandClient Audit log : CEF Formatted syslog appender`, and `Sideband Audit log : CEF Formatted syslog appender` sections.

   > **Note**
   >
   > Each `Socket` appender is followed by two related appenders, `RollingFile` and `PingFailover`. Together, they create a running `audit-cef-syslog-failover.log` file in the *<PA_HOME>*/log/pingaccess.log directory if CEF logging fails for any reason. If you uncomment the `Socket` appenders, make sure to uncomment the related appenders also.

   Example:

In the `Api Audit log : CEF Formatted syslog appender` section, uncomment the `ApiAuditLogToCEF-Syslog Socket` appender configuration:

```
<!--
<Socket name="ApiAuditLogToCEF-Syslog" host="{syslog.host}" port="{syslog.port}"
protocol="{syslog.protocol}" ignoreExceptions="false">
   <PingSyslogLayout>
      <PatternLayout>
         <pattern>%escape{CEF}{CEF:0|Ping Identity|PingAccess|%X{AUDIT.paVersion}|
%X{exchangeId}|API_AccessEvent|0|rt=%d{ISO8601} msg=%X{AUDIT.responseCode}
duid=%X{AUDIT.subject} src=%X{AUDIT.client} requestMethod=%X{AUDIT.method}
request=%X{AUDIT.requestUri} cs1Label=AuthenticationMechanism cs1=%X{AUDIT.authMech}
cs2Label=RoundTripMS cs2=%X{AUDIT.roundTripMS} externalId=%X{AUDIT.trackingId} %n}</pattern>
      </PatternLayout>
   </PingSyslogLayout>
</Socket>

<RollingFile name="ApiAuditLogToCEF-Syslog-FILE"
fileName="${sys:pa.home}/log/pingaccess_api_audit_cef_syslog_failover.log"
filePattern="${sys:pa.home}/log/pingaccess_api_audit_cef_syslog_failover.%d{yyyy-MM-dd}.log"
ignoreExceptions="false">
   <PatternLayout>
      <pattern>%escape{CEF}{CEF:0|Ping Identity|PingAccess|%X{AUDIT.paVersion}|%X{exchangeId}|
API_AccessEvent|0|rt=%d{ISO8601} msg=%X{AUDIT.responseCode} duid=%X{AUDIT.subject}
src=%X{AUDIT.client} requestMethod=%X{AUDIT.method} request=%X{AUDIT.requestUri}
cs1Label=AuthenticationMechanism cs1=%X{AUDIT.authMech} cs2Label=RoundTripMS
cs2=%X{AUDIT.roundTripMS} externalId=%X{AUDIT.trackingId} %n}</pattern>
   </PatternLayout>
   <Policies>
      <TimeBasedTriggeringPolicy />
   </Policies>
</RollingFile>

<PingAccessFailover name="ApiAuditLogToCEF-Syslog-Failover" primary="ApiAuditLogToCEF-Syslog"
error="File">
   <Failovers>
      <AppenderRef ref="ApiAuditLogToCEF-Syslog-FILE" />
   </Failovers>
</PingAccessFailover>
-->
```

Repeat this with the `EngineAuditLogToCEF-Syslog`, `AgentAuditLogToCEF-Syslog`, `SidebandClientAuditLogToCEF-Syslog`, and `SidebandAuditLogToCEF-Syslog` appenders.

3. In the `ApiAuditToCEF-Syslog`, `EngineAuditToCEF-Syslog`, `AgentAuditToCEF-Syslog`, `SidebandClientAuditToCEF-Syslog`, and `SidebandAuditToCEF-Syslog Socket` appenders, replace the following placeholder parameter values:

syslog.host

The URL of your syslog host server.

syslog.port

The port that your syslog host server uses.

syslog.protocol

The protocol that your syslog host server uses. Valid values are UDP or TCP.

> **Note**
>
> Only the TCP protocol supports failover.

4. Save and close the file.

Writing logs in JSON format

You can configure PingAccess to write logs in JavaScript Object Notation (JSON) format using the log4j2 logging library. JSON is a common logging format for security information and event management (SIEM) tracking systems and is easily human-readable.

About this task

PingAccess includes JSON log templates, which you can find in the `<PA_HOME>/conf/log4j/json-templates` directory, for the following log files:

- `pingaccess.log`
- `pingaccess_api_audit.log`
- `pingaccess_agent_audit.log`
- `pingaccess_engine_audit.log`
- `pingaccess_sideband_audit.log`
- `pingaccess_sideband_client_audit.log`

> **Note**
>
> The `jvm-garbage-collection.log` file isn't log4j2-enabled, so PingAccess doesn't provide a JSON log template for it.

> **Important**
>
> The `log4j2.xml` file contains `RollingFile` appenders that produce output in both standard and JSON format. By default, the appenders for both formats output to the same filename.
>
> - To output only one format, comment out the appender reference for the other format. Learn more in step 3.
> - To output logs in both standard and JSON format, configure different filenames for each format in the `RollingFile` appender configurations. Otherwise, both formats display in the same file. Learn more in step 4.

Steps

1. Open the `<PA_HOME>/conf/log4j2.xml` file in a text editor.
2. Uncomment the JSON appender references in the root, `apiaudit`, `engineaudit`, `agentaudit`, `sidebandclientaudit`, and `sidebandaudit` logger configurations.

   Example:

   In the `Set up the Root logger` section of the `log4j2.xml` file, uncomment the `File-JSON` appender reference:

   ```
   <!-- ======================== -->
   <!-- Set up the Root logger  -->
   <!-- ======================== -->
   <AsyncRoot level="INFO" includeLocation="false">
       <AppenderRef ref="File"/>
       <AppenderRef ref="File-JSON" />
       <!--<AppenderRef ref="CONSOLE" />-->
       <!--<AppenderRef ref="CONSOLE-JSON" />-->
       <!--<AppenderRef ref="SYSLOG" />-->
   </AsyncRoot>
   ```

   Repeat this in the `Audit log configuration` section with the `ApiAuditLog-JSON`, `EngineAuditLog-JSON`, `AgentAuditLog-JSON`, `SidebandClientAuditLog-JSON`, and `SidebandAuditLog-JSON` appender references.

   > **Note**
   >
   > If you want to write the logs to the console instead of, or in addition to a file, uncomment the `CONSOLE-JSON`, `CONSOLE-ApiAuditLog-JSON`, `CONSOLE-EngineAuditLog-JSON`, `CONSOLE-AgentAuditLog-JSON`, `CONSOLE-SidebandClientAuditLog-JSON`, and `CONSOLE-SidebandAuditLog-JSON` appender references as necessary.

3. If you want JSON output only, comment out the appender references for the non-JSON format output.

   > **Note**
   >
   > Doing so prevents PingAccess from writing both standard and JSON formats to the same log file.

   Example:

   In the `Set up the Root logger` section of the `log4j2.xml` file, comment out the `File` appender reference:

   ```
   <!-- ======================== -->
   <!-- Set up the Root logger  -->
   <!-- ======================== -->
   <AsyncRoot level="INFO" includeLocation="false">
       <!--<AppenderRef ref="File"/>-->
       <AppenderRef ref="File-JSON" />
       <!--<AppenderRef ref="CONSOLE" />-->
       <!--<AppenderRef ref="CONSOLE-JSON" />-->
       <!--<AppenderRef ref="SYSLOG" />-->
   </AsyncRoot>
   ```

Repeat this in the `Audit log configuration` section with the `ApiAuditLog-File`, `EngineAuditLog-File`, `AgentAuditLog-File`, `SidebandClientAuditLog-File`, and `SidebandAuditLog-File` appender references.

4. If you want to output two separate log files for standard and JSON format, change the name of the output file in the `RollingFile` appender configurations for the JSON format.

   Example:

   In the `API auditing file logging configuration` section, go to the `ApiAuditLog-JSON` `RollingFile` appender configuration. Modify the `fileName` and `filePattern`:

   ```
   <!-- API Audit log : JSON format file logging configuration -->
           <RollingFile name="ApiAuditLog-JSON"
                        fileName="${sys:pa.home}/log/pingaccess_json_api_audit.log"
                        filePattern="${sys:pa.home}/log/pingaccess_json_api_audit.%d{yyyy-MM-dd}.log"
                        ignoreExceptions="false">
               <JsonTemplateLayout eventTemplateUri="${sys:pa.log4j.json.templates.uri}/api-audit-
   log.json"/>
               <Policies>
                   <TimeBasedTriggeringPolicy />
               </Policies>
           </RollingFile>
   ```

   Repeat this with the `EngineAuditLog-JSON`, `AgentAuditLog-JSON`, `SidebandAuditClientLog-JSON`, and `SidebandAuditLog-JSON` appender configurations.

5. (Optional) For each `JsonTemplateLayout` value, designate the URI location of the desired JSON templates.

   > **Note**
   >
   > The `${sys:pa.log4j.json.templates.uri}` URI designates the default location where the JSON log file templates are stored. You can replace this with a custom URI filepath. Otherwise, log files are stored in their default location of `<PA_HOME>/conf/log4j/json-templates`.

   Using the same example from the previous step, you can find `JsonTemplateLayout` after the `RollingFile` details:

   ```
   <!-- API Audit log : JSON format file logging configuration -->
           <RollingFile name="ApiAuditLog-JSON"
                        fileName="${sys:pa.home}/log/pingaccess_json_api_audit.log"
                        filePattern="${sys:pa.home}/log/pingaccess_json_api_audit.%d{yyyy-MM-dd}.log"
                        ignoreExceptions="false">
               <JsonTemplateLayout eventTemplateUri="${sys:pa.log4j.json.templates.uri}/api-audit-
   log.json"/>
               <Policies>
                   <TimeBasedTriggeringPolicy />
               </Policies>
           </RollingFile>
   ```

6. Save and close the `log4j2.xml` file.

Custom log patterns

To create custom log patterns in log4j2-enabled logs using JSON format, you must use special syntax.

For example, if a log file appender references a custom HTTP header using `%clientrequestheader` to log `x-myheader`:

```
<RollingFile ... >
<PatternLayout>
<pattern>%d | %header{x-myheader} | %m%n</pattern>
</PatternLayout>
...
</RollingFile>
```

In the corresponding JSON template (for example, `api-audit-log.json`), you must refer to the `%clientRequestHeader{x-myheader}` using the following JSON object:

```
"myheader": {
    "$resolver": "pattern",
    "pattern": "%clientRequestHeader{x-myheader}"
},
```

Note

You can also use `MDC` as the `$resolver` to get the value directly. For example:

```
"myheader": {
    "$resolver": "mdc",
    "key": "AUDIT.http-client-request-header-x-myheader",
    "stringified": true
},
```

If you want to use `MDC` but aren't sure what options are available, you can print out all available values in MDC:

```
"allvalues": {
    "$resolver": "mdc",
    "stringified": true
},
```

You can find the reference to the relevant JSON template in the `RollingFile` appender configurations in the `log4j2.xml` file. The JSON file appender names include a `-JSON` suffix. The associated `eventTemplateUri` value indicates the relevant JSON template name.

```
<RollingFile name="ApiAuditLog-JSON"
             fileName="${sys:pa.home}/log/pingaccess_api_audit.log"
             filePattern="${sys:pa.home}/log/pingaccess_api_audit.%d{yyyy-MM-dd}.log"
             ignoreExceptions="false">
     <JsonTemplateLayout eventTemplateUri="${sys:pa.log4j.json.templates.uri}/api-audit-log.json"/>
     <Policies>
         <TimeBasedTriggeringPolicy />
     </Policies>
</RollingFile>
```

Customize and localize PingAccess

This section contains information on how to customize PingAccess templates and localize user-facing messages.

- For information on customizing PingAccess page templates and on the difference between customizable templates and system templates, see User-facing page customization reference.
- For information on localizing user-facing system status messages, see User-facing page localization reference.

User-facing page customization reference

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 application programming interface (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, see 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.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

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.

Customizable page templates

At runtime, the user's browser is directed to the appropriate page, depending on the operation being performed and where the related condition occurs. For example, if 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.
`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.

Template File Name	Purpose	Type	Action
`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.

System Templates

The templates stored in `<PA_HOME>/conf/template/system` are system templates. Do not modify these templates directly unless directed by Ping. This table shows the purpose and associated action, if any, for each of these files.

File Name	Purpose	Type	Action
`admin.loggedout.page.template.html`	Displayed when a user completes a single logout (SLO) initiated from the PingAccess admin console.	Normal	The user's session at the identity provider (IdP) and the PingAccess administrative console has been terminated.
`agent.bootstrap.template.properties`	Used to generate the `agent.properties` file for an agent.	Normal	None
`engine.bootstrap.template.properties`	Used to generate the `bootstrap.properties` file for an engine.	Normal	None
`fragment.preservation.request.html`	Used to preserve the fragment from the requested Uniform Resource Locator (URL) in client-side storage during a PingAccess OpenID Connect (OIDC) sign-on flow.	Normal	None
`fragment.preservation.response.html`	Used to restore the fragment from client-side storage for the originally requested URL when a PingAccess OIDC sign-on flow has completed.	Normal	None
`invalid.token.json`	Used to challenge a user agent for authentication when the user-agent specifies an Accept header field containing `application/json`.	Normal	The user agent interacts with the end user to obtain an OAuth token.

File Name	Purpose	Type	Action
`post.preservation.request.html`	Used to preserve the HTML form data from a POST request in client-side storage during a PingAccess OIDC sign-on flow.	Normal	None
`post.preservation.response.encoded.html`	Used to submit encrypted HTML form data to PingAccess from a previously preserved POST request when a PingAccess OIDC sign-on flow completes.	Normal	None
`post.preservation.response.html`	Used to reconstruct an HTML form to resubmit restored POST data when a PingAccess OIDC sign-on flow completes.	Normal	None
`redirect.response.html`	Used to redirect a browser to the token provider for authentication.	Normal	None
`replica.bootstrap.template.properties`	Used to generate the bootstrap.properties file for a replica admin.	Normal	None
`site.authenticator.rst.xml`	Used to produce a request to send to the PingFederate Security Token Service (STS) endpoint to exchange a PingAccess cookie or OAuth token for a Web Access Management (WAM) token.	Normal	None
`unauthorized.response.html`	Used to produce a challenge for authentication to an OAuth client running in a browser-based application.	Normal	None

User-facing page localization reference

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. 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.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

Managing Federal Information Processing Standards (FIPS) mode

Federal Information Processing Standards (FIPS) mode ensures that PingAccess uses encryption algorithms that meet FIPS requirements.

If FIPS mode is enabled, you can view your environment's FIPS mode status in the PingAccess admin console or audit log:

- To view FIPS mode in the admin console, go to **Account > About** and in the **System Information** section, find FIPS mode status.
- To view FIPS mode in the audit log, review the audit log after starting PingAccess. If FIPS mode is enabled, an info-level entry indicates this status. For example:

```
INFO [] Fipsconfig - PingAccess is currently running in FIPS Mode.
```

Some features of PingAccess operate differently or are unavailable in FIPS mode.

Features that aren't supported in FIPS mode:

- SHA-1
- PKCS#12 certificates and private keys

Certificate and private key format requirements:

- PingAccess only supports PEM-formatted certificates and private keys, meaning:
  - You can only import or export key pairs using the PEM-encoded format.
  - PingAccess only accepts PBES2 and AES or Triple DES encryption.
  - PingAccess requires 128-bit salt.

  > **Note**
  >
  > In practice, this could mean that you can only import PEM files generated by PingFederate.

- For PEM files, the private key must precede the certificates.

Password format requirements:

- The password must contain at least 14 characters.

To manage FIPS mode, select a tab.

Enabling FIPS mode

Enabling FIPS mode
About this task

Enable FIPS mode to ensure that PingAccess exclusively uses encryption algorithms permitted by the FIPS standard. If your environment is clustered, make sure to perform this procedure on all nodes.

> **Note**
>
> In this procedure, you can manually specify security providers, TLS protocols, and TLS cipher suites that can be used. If your manual inclusions are not FIPS-compliant, your environment might not be FIPS-compliant even in FIPS mode.

Steps

1. Open the `<PA Home>/conf/fips-mode.properties` file or create it if it's been removed.
2. Set the `pa.fips.mode` property to `true`.

   Example:

   ```
   pa.fips.mode=true
   ```

3. (Optional) Exempt one or more security providers from being excluded by FIPS mode by adding a comma-separated list of class names to the `pa.fips.additionalAllowedProviders` property.

   Example:

   ```
   pa.fips.additionalallowedproviders=X,Y
   ```

4. (Optional) Add or remove TLS protocols by editing the `pa.fips.tls.protocols` property to include a comma-separated list of valid TLS protocols.

   The default is:

   ```
   pa.fips.tls.protocols = TLSv1.2
   ```

5. (Optional) Add or remove TLS cipher suites by editing the `pa.fips.tls.ciphers` property to include a comma-separated list of valid TLS cipher suites.

The default is:

```
pa.fips.tls.ciphers = TLS_AES_256_GCM_SHA384, \
                      TLS_AES_128_GCM_SHA256, \
                      TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, \
                      TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, \
                      TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, \
                      TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, \
                      TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, \
                      TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, \
                      TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, \
                      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, \
                      TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384, \
                      TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384, \
                      TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256, \
                      TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256, \
                      TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384, \
                      TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384, \
                      TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256, \
                      TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256, \
                      TLS_EMPTY_RENEGOTIATION_INFO_SCSV
```

Note

Some of the default cipher suites aren't supported by every JDK version that can be used with PingAccess. If a TLS cipher suite isn't supported by the JDK version you're using, PingAccess will log a warning in the `pingaccess.log` file when the cipher suite is invoked.
PingAccess can ignore any flagged TLS cipher suites with no performance impact. To clear the warning message, you can remove the flagged suites from the `pa.fips.tls.ciphers` property.

6. Save and close the `<PA Home>/conf/fips-mode.properties` file.
7. If you're running PingAccess as a Windows service, reconfigure the classpath for the libraries required for FIPS mode:
    1. Comment out the following line:

        ```
        set.default.BC_PATH=../../resource/bc/non-fips
        ```

    2. Uncomment the following line or set a `BC_PATH` environment variable to `../../resource/bc/fips`:

        ```
        # set.default.BC_PATH=../../resource/bc/fips
        ```

    Note

    Learn more in the inline comments in the `<PA_HOME>/sbin/windows/PingAccessService.conf` file.

8. Restart PingAccess.

Disabling FIPS mode

Disabling FIPS Mode
About this task

Disable FIPS Mode to allow the use of non-FIPS compliant encryption. If your environment is clustered, perform this procedure on all nodes.

Steps

1. Open the `<PA Home>/conf/fips-mode.properties` file.
2. Set the `pa.fips.mode` property to `false`.

   ```
   pa.fips.mode=false
   ```

3. Save and close the `<PA Home>/conf/fips-mode.properties` file.
4. If you're running PingAccess as a windows service, reconfigure the classpath for the libraries required for FIPS mode:
   1. Uncomment the following line:

      ```
      # set.default.BC_PATH=../../resource/bc/non-fips
      ```

   2. Comment out the following line or the `BC_PATH` environment variable to `../../resource/bc/fips` that you set:

      ```
      set.default.BC_PATH=../../resource/bc/fips
      ```

   Note

   Learn more in the inline comments in the `<PA_HOME>/sbin/windows/PingAccessService.conf` file.

5. Restart PingAccess.

Configuring PingAccess to use Amazon Key Management Services

During initial startup, PingAccess automatically generates a randomized master key, which by default is not encrypted. If you are running in Amazon Web Services (AWS), you can configure PingAccess to use Amazon Key Management Services (KMS) to encrypt the master key.

Before you begin

- Make sure that you have an active connection to AWS.
- Use AWS KMS to generate a key to use for the PingAccess master key encryption.

> **Note**
>
> For more information about managing access rights to your keys using key policies or AWS Identity and Access Management (IAM), see AWS Key Management Service.

About this task

To configure the encryption of the PingAccess master key, modify the `pa.jwk.properties` file found in `<PA_HOME>/conf`.

Steps

1. Stop PingAccess.
2. In a text editor, open `<PA_HOME>/conf/pa.jwk.properties`.
3. Locate the `pa.hostkey.masterKeyEncryptor` property .
4. Enable master key encryption.
    1. Change `com.pingidentity.pa.crypto.NoOpMasterKeyEncryptor` to the AWS KMS master key encryptor class name `com.pingidentity.pingcommons.aws.key.AwsKmsMasterKeyEncryptor`.
    2. Locate the ID for the key that you generated using AWS KMS.
    3. If this is not the first time starting PingAccess, prefix the key ID with `“ENCRYPT:”`.

       Example:

       After making changes, the properties file should look similar to the following:

       ```
       pa.hostkey.masterKeyEncryptor=com.pingidentity.pingcommons.aws.key.AwsKmsMasterKeyEncryptor

       pa.hostkey.keyId=ENCRYPT:d4e6adab-e20c-4339-ba76-e4cb1348713f
       ```
5. Save and close the updated `pa.jwk.properties` file.
6. Restart PingAccess.

   The PingAccess master file `pa.jwk` is encrypted using Amazon KMS.

Use environment variables to override configuration settings

To change a PingAccess server’s configuration, you can use environment variables to override the settings in multiple configuration files. This eliminates the need to directly modify each of those files.

Environment variables simplify the process of container management because you can make all of your configuration changes in one place.

> **Important**
>
> If you’re running PingAccess in a clustered deployment, you must apply environment variable changes to each cluster node individually. PingAccess can’t replicate property files, and by extension, environment variables, from the administrative node.

When PingAccess starts, it overrides property values in the configuration files with the values of the environment variables. During startup, PingAccess also logs environment variables that start with `PA_` in the `pingaccess.log` file. Some initialization items might go to `system.out`.

> **Important**
>
> If a utility uses a configuration file that's been modified by an environment variable, the utility also inherits any changes made by that environment variable.
> For example, the `obfuscate.sh` script uses properties from the `pa.jwk.properties` file. If you create an environment variable that overrides a property in that file, `obfuscate.sh` also uses that modified property value.

To set your environment variables, use a deployment tool of your choice, such as Kubernetes or Docker Compose. Use the format `PA_<FILE_NAME_WITHOUT_EXTENSION>_<PROPERTY_NAME>`, where:

- `PA` is short for PingAccess. You must begin all environment variables with this phrase.
- *<FILE_NAME_WITHOUT_EXTENSION>* is the name of the file containing the property that you want to modify. Leave out the file extension.
- *<PROPERTY_NAME>* is the name of the property that you want to modify.

When naming an environment variable:

- Replace any periods in the file or property name with a single underscore.
    - Replace any dashes with two underscores.
- Only use letters and underscores. Leave out any separators, such as parenthesis, braces, brackets, commas, or semicolons.
- Write the whole environment variable in upper case.

> **Note**
>
> Make sure that you use consistent case and spelling when setting environment variables to avoid unexpected behavior. If you have multiple references to the same property but case varies between those references, PingAccess can only use one version.

- The environment variable for the `pa.hostkey.keyId` property in the `pa.jwk.properties` file is `PA_PA_JWK_PA_HOSTKEY_KEYID`.
- The environment variable for the `maxIdle` property in the `log4j2.db.properties` file is `PA_LOG4J2_DB_MAXIDLE`.
- The environment variable for the `admin.header.X-Content-Type-Options` property in the `run.properties` file is `PA_RUN_ADMIN_HEADER_XCONTENTTYPE__OPTIONS`.
- The environment variable for the `engine.httptransport.socketTimeout` property in the `run.properties` file is `PA_RUN_ENGINE_HTTPTRANSPORT_SOCKETTIMEOUT`.

You can override the value of any property defined in one of the property files from the `<pa>/conf` directory. These property files include:

- `engine-registration.properties`
- `fips-mode.properties`

- `log4j2.db.properties`
- `pa.jwk.properties`
- `run.properties`

> **Note**
>
> It was already possible to override settings in the `engine-registration.properties` file with the `ENGINE_NAME` environment variable. You can continue to use the `ENGINE_NAME` environment variable, or you can update all instances to the new format for consistency.

Don't create environment variables for properties defined in:

- Files outside of the `/conf` directory
- `.properties` files from the `/conf/localization` directory.

Reference Guides

This section contains reference guides for common tasks related to making the most of your PingAccess environment and extending its capabilities.

- For information on enabling external applications to communicate with PingAccess, see PingAccess API endpoints.
- For information on setting up a clustered PingAccess environment, see Clustering in PingAccess.
- For a list of configurable parameters that PingAccess uses at runtime, see the Configuration file reference.
- For help deciding what type of PingAccess deployment best suits your environment, see the PingAccess deployment guide.
- For information on extending the functionality of PingAccess rules in your environment, see Groovy in PingAccess.
- For information on optimizing your PingAccess environment, see Performance tuning.

PingAccess API endpoints

The following 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

Enables 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 (OAuth AS) to interface with PingAccess as an OAuth resource server.

Administrative API endpoints

Enables 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 default application reserved path, `/pa`. You can modify the reserved path using the PingAccess Admin API. If you do, you must update endpoint and other applicable application URLs accordingly.
Similarly, if you select the **Use context root as reserved resource base path** check box on your PingAccess application, you must enter the context root of the application before the reserved path in endpoint and other applicable application URLs. For example, if the context root of your application is `myApp` and you haven't modified the reserved path, use the path `myApp/pa` instead of `/pa`.

The features documented here are affected by the settings in the configuration file. For more information, see the Configuration file reference.

Heartbeat endpoint

The heartbeat endpoint verifies that the PingAccess server is running and, depending on security settings, displays 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 responds to this endpoint on port 9000, and a clustered engine responds 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.

> **Note**
>
> The Uniform Resource Locator (URL) should begin with the server name and the PingAccess runtime port number. For example, `https://hostname:3000/pa/heartbeat.ping`.
> If you selected the **Use context root as reserved resource base path** check box on your PingAccess application, this feature creates an instance of any reserved PingAccess resources under the application's context root. As such, the context root of the application needs to prepend the reserved context application root (`/pa` by default) in any file paths that reference it.
> If the context root of your application is `myApp`, the path to the heartbeat endpoint would be `myApp/pa/heartbeat.ping` and the URL would be `https://hostname:3000/myApp/pa/heartbeat.ping` instead.

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
`$monitor.getTotalJvmMemory('bytes'|'KB'|'MB'|'GB')`	Returns the total memory in the Java Virtual Machine (JVM). Enter `'bytes'`, `'KB'`, `'MB'`, or `'GB'` to specify the units. If you don't specify the units, `'bytes'` is the default value.
`$monitor.getUsedJvmMemory('bytes'|'KB'|'MB'|'GB')`	Returns the used memory in the JVM. Enter `'bytes'`, `'KB'`, `'MB'`, or `'GB'` to specify the units. If you don't specify the units, `'bytes'` is the default value.
`$monitor.getFreeJvmMemory('bytes'|'KB'|'MB'|'GB')`	Returns the free memory in the JVM. Enter `'bytes'`, `'KB'`, `'MB'`, or `'GB'` to specify the units. If you don't specify the units, `'bytes'` is the default value.

Value	Description
`$monitor.getTotalPhysicalSystemMemory('bytes'|'KB'|'MB'|'GB')`	Returns the total system memory. Enter `'bytes'`, `'KB'`, `'MB'`, or `'GB'` to specify the units. If you don't specify the units, `'bytes'` is the default value.
`$monitor.getTotalUsedPhysicalSystemMemory('bytes'|'KB'|'MB'|'GB')`	Returns the used system memory. Enter `'bytes'`, `'KB'`, `'MB'`, or `'GB'` to specify the units. If you don't specify the units, `'bytes'` is the default value.
`$monitor.getTotalFreePhysicalSystemMemory('bytes'|'KB'|'MB'|'GB')`	Returns the free system memory. Enter `'bytes'`, `'KB'`, `'MB'`, or `'GB'` to specify the units. If you don't specify the units, `'bytes'` is the default value.
`$monitor.getHostname()`	Returns the host name 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 cause the value to be 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.

Value	Description
`$monitor.getNumberOfSites()`	Returns the current number of configured sites in the PingAccess configuration database. In a clustered environment, on the engine nodes, this number will reflect the number of sites associated with applications rather than the number of configured sites that show on the admin node. For more information, see the Clustering in PingAccess documentation. This value is not included in the default template but can be added by the system administrator if desired.
`$monitor.getLastRefreshTime('yyyy/MM/dd HH:mm:ss')`	Returns the time that 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 for the new value to take effect.

Calls to this endpoint can be logged in the audit log. You can enable heartbeat call logging using the `/httpConfig/monitoring` administrative endpoint. For more information, see Administrative API endpoints.

OpenID Connect endpoints

Specific endpoints are 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 the `<PA_HOME>/conf/run.properties` file.

> **Note**
>
> If you selected the **Use context root as reserved resource base path** check box on your PingAccess application, this feature creates an instance of any reserved PingAccess resources under the application's context root. As such, the context root of the application needs to prepend the reserved context application root (`/pa` by default) in any file paths that reference it.
> If the context root of your application is `myApp` , the paths to the OIDC endpoints would be:
>
> - `/myApp/pa/oidc/logout`
> - `/myApp/pa/oidc/cb`
> - `/myApp/pa/oidc/JWKS`
> - `/myApp/pa/oidc/logout.png`

/pa/oidc/logout

The `pa/oidc/logout` endpoint clears the browser cookie containing the PingAccess token. This enables end users to trigger the removal of their own PingAccess cookie from the browser that they're using, which redirects them to the logged out page.

You can modify the logged out page template in the `<PA_INSTALL>/conf/template/general.loggedout.page.template.html` file.

> **Note**
>
> This endpoint does not retain any server-side state to denote log off. Additionally, unless single logout (SLO) is selected for the token provider, this endpoint clears the cookie only from the requested host or domain. This means that the cookie might still exist in requests bound for other hosts or domains.
> If you selected the **Use Single-Logout** option when configuring the token provider, this endpoint also sends a logout request to the token provider, which completes a full SLO flow.

/pa/oidc/cb

The `/pa/oidc/cb` endpoint, along with the application virtual host, becomes the redirect Uniform Resource Identifier (URI) for the token provider configuration on the client.

/pa/oidc/JWKS

The `/pa/oidc/JWKS` endpoint is used by the token provider's JSON Web Token (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 Configuring JSON token management.

/pa/oidc/logout.png

The `/pa/oidc/logout.png` endpoint is used by the token provider to initiate a logout from PingAccess in conjunction with SLO functionality, terminating the PingAccess tokens across domains.

Authentication Token Management endpoint

This page describes the endpoint used to validate JSON Web Tokens.

/pa/authtoken/JWKS

Backend sites use the Authentication Token Management endpoint to validate the signature of a JSON Web Token (JWT).

> **Note**
>
> If you selected the **Use context root as reserved resource base path** check box on your PingAccess application, this feature creates an instance of any reserved PingAccess resources under the application's context root. As such, the context root of the application needs to prepend the reserved context application root (`/pa` by default) in any file paths that reference it.
> If the context root of your application is `myApp` , the path to the Authentication Token Management endpoint would be `myApp/pa/authtoken/JWKS` instead.

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

An OAuth authorization server uses this endpoint to acquire PingAccess public keys to encrypt access tokens. The output uses the Internet Engineering Task Force (IETF) JSON Web Token (JWT) format for public keys.

> **Note**
>
> If you selected the **Use context root as reserved resource base path** check box on your PingAccess application, this feature creates an instance of any reserved PingAccess resources under the application's context root. As such, the context root of the application needs to prepend the reserved context application root (`/pa` by default) in any file paths that reference it.
> If the context root of your application is `myApp` , the path to the OAuth endpoint would be `myApp/pa/oauth/JWKS` instead.

Administrative API endpoints

PingAccess ships with interactive documentation for both developers and non-developers to explore the PingAccess application programming interface (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 the `/pa-admin-api/v3/api-docs/<PA_HOME>/conf/run.properties` file. For example, https://*<PA_HOME>*:*<PORT>*/pa-admin-api/v3/api-docs/.

> **Note**
>
> If you selected the **Use context root as reserved resource base path** check box in your PingAccess application, this feature creates an instance of any reserved PingAccess resources under the application's context root. As such, the context root of the application needs to prepend the reserved context application root (`/pa` by default) in any file paths that reference it. If the context root of your application is `myApp` , the file path would start with `/myApp/pa` .

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

Admin API documentation Swagger-UI specifications

The Swagger-UI component that displays the PingAccess admin API documentation uses OpenAPI specification (OAS) 2.0.

> **Important**
>
> The specification that the PingAccess admin API docs used previously, Swagger 1.2, has been deprecated. The Swagger 1.2 specification is still available at https://<*PA_HOME*>:<*PORT*>/pa-admin-api/v3/api-docs/pa/api-docs.json but might be removed from future versions of PingAccess.

You can find the PingAccess admin API's OAS 2.0 specifications at either of the following:

- https://<*PA_HOME*>:<*PORT*>/pa-admin-api/v3/api-docs/pa/api-docs-v2.json
- https://<*PA_HOME*>:<*PORT*>/pa-admin-api/v3/api-docs/pa/api-docs-v2.yaml

> **Tip**
>
> Access to these specifications simplifies the process of integrating the PingAccess admin API with modern API clients, such as Postman.

Clustering in PingAccess

PingAccess provides clustering features that allow a group of PingAccess servers to appear as a single system.

Server clustering can facilitate high availability of critical services and can also increase performance and overall system throughput. However, availability and performance are often at opposite ends of the deployment spectrum. You might need to make some configuration tradeoffs that balance availability with performance to accommodate specific deployment goals.

> **Note**
>
> Settings in the configuration file could affect the features documented here. For more information, see the Configuration file reference guide.

Components of a PingAccess Cluster

PingAccess clusters are made up of three types of nodes:

The Administrative Node

Provides the administrator with a configuration interface.

The Replica Administrative Node

Provides the administrator with the ability to recover a failed administrative node using a manual failover procedure. For more information, see Manually promoting the replica administrative node.

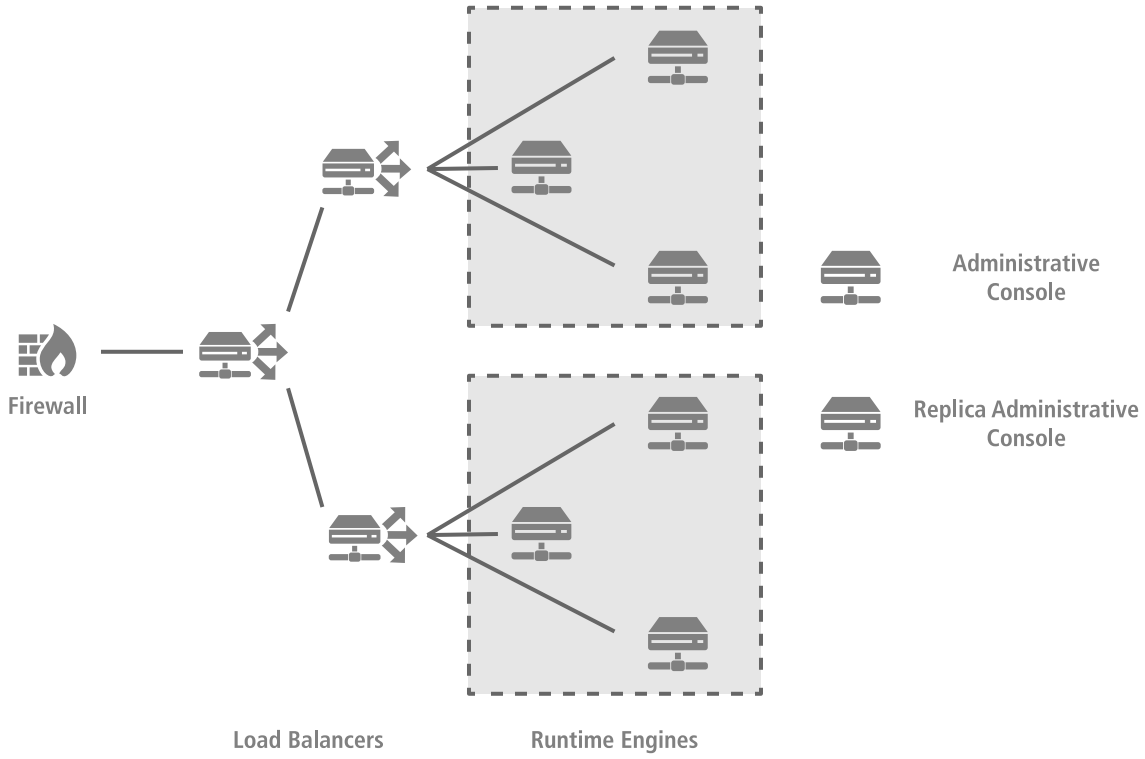
The Engine Nodes

Handle incoming client requests and evaluate policy decisions based on the configuration replicated from the administrative node.

> **Note**
>
> You can configure any number of engine nodes in a cluster, but you can configure only one administrative node and one replica administrative node in a cluster. State information is not shared between engine nodes.

Configuration information from the administrative console and API is replicated to all of the engine nodes and the replica administrative node from the administrative node, as is the license file on the administrative node. Engine nodes do not require a license to function, but some default templates look different depending on the information in the license.

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.



Node failure implications

Node failure within a PingAccess cluster can have short-term or long-term implications for your environment, depending on the state of your network and the type of node or nodes that failed. The following table describes some common node issues and recommends what kind of action to take.

Node issue	Result	Recommendation
Administrative node failure	The engine nodes can function using their stored configurations but cannot update their configurations.	Fail over to the replica administrative node until the administrative node can be restarted.

Node issue	Result	Recommendation
Replica administrative node failure	The engine nodes and administrative node can function normally, but you won't be able to fail over to the replica administrative node if something happens to the administrative node.	Restart the replica administrative node as soon as possible.
Administrative and replica node failure	The engine nodes can function using their stored configurations, but cannot update their configurations. No failover option is available.	Restart the administrative node as soon as possible, or restart the replica administrative node and fail over to it.
One or more engine nodes cannot reach the administrative node	Affected engine nodes can function using their 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 access to the administrative node as soon as possible.

Cluster properties

> **Note**
>
> 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 clustered PingAccess engine node 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.

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

bootstrap.properties

Property	Description
`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.

Property	Description
`engine.admin.configuration.bootstrap.truststore`	Defines the trust store, in JWK format, that is used for communication with the administrative console.

> **Note**
>
> You can tune the cache using the EHCache Configuration Properties, `pa.ehcache.*`, listed in the Configuration file reference guide.

Cluster node status

The administrative console provides two important visual elements which communicate the current status of the replica administrative node and the engine nodes:

- A status indicator, which communicates whether the node is healthy.
- A **Last Updated** field, which communicates the date and time that the node was last updated.

You can find this information on the **Administrative Nodes** page and the **Engines** page.

Status indicators use the value for *<admin.polling.delay>* as an interval to measure node health. A node's status can be green (good status), yellow (degraded status), or red (failed status):

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

PingAccess binds to all network interfaces by default to support routing traffic over multiple network interfaces. The default bind address PingAccess uses is 0.0.0.0. To prevent PingAccess from binding to all network interfaces, you can edit one or more of the following parameters in the `conf/run.properties` file:

```
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
```

Specify a new bind address for the parameter that you want to modify.

Configuring a PingAccess cluster

Install and configure PingAccess on each node in a cluster, including the administrative node, a replica administrative node, and one or more engine nodes.

About this task

The initial node you configure becomes the administrative node, which you will use to configure the rest of the cluster.

> **Important**
>
> Setting the `pa.operational.mode` property on each node is part of the configuration process. Do not modify this property until directed to do so.

Steps

1. Install PingAccess on each cluster node.
2. Configure the administrative node:
    1. Open the `conf/run.properties` file in a text editor and change the `pa.operational.mode` value to `CLUSTERED_CONSOLE`.

       This property is case-sensitive.
    2. Start PingAccess.
    3. Follow steps 1-14 of Generating new key pairs to create a new key pair for the CONFIG QUERY listener. Make the following adjustments to steps 4-5:
        1. To complete step 4, enter the DNS name of the administrative node in the **Common Name** field.
        2. To complete step 5, enter both the DNS name of the replica administrative node and the DNS name of the administrative node in the **Subject Alternative Names** field. Alternately, configure the **Subject Alternative Names** field as a wildcard certificate.

           > **Note**
           >
           > You can use an Internet Protocol (IP) address as the common name or in the **Subject Alternative Names** field, as long as those values are used in the administrative node fields on the **Administrative Nodes** configuration page.

           > **Note**
           >
           > You will need this key pair in step 3a to set up the replica administrative console.
    4. Follow steps 1-4 of Assigning key pairs to HTTPS listeners to assign the key pair you just created to the CONFIG QUERY listener.
    5. Follow steps 1-6 in Configuring administrative nodes to configure the administrative node settings, then review the *What to do next* section. Make the following adjustment to step 2:
        1. To complete step 2, define the primary administrative node as a `host:port` pair in the **Host** field.

> **Note**
>
> The host you specify must be a resolvable DNS name for the node or the node's IP address. The port must be the TCP port that PingAccess listens to for the administrative interface. By default, this port is 9090.

6. Follow steps 1-14 of Generating new key pairs to create a new key pair for the ADMIN listener. Make the following adjustments to steps 4-5:
    1. To complete step 4, enter the DNS name of the administrative node in the **Common Name** field.
    2. To complete step 5, enter both the DNS name of the replica administrative node and the DNS name of the administrative node in the **Subject Alternative Names** field. Alternately, configure the **Subject Alternative Names** field as a wildcard certificate.

        > **Note**
        >
        > You can use an IP address as the common name or in the **Subject Alternative Names** field as long as those values are used in the administrative node fields on the **Administrative Nodes** configuration page.

7. Follow steps 1-4 of Assigning key pairs to HTTPS listeners to assign the key pair you just created to the ADMIN listener.
8. Restart PingAccess.

3. Configure the replica administrative node.

> **Note**
>
> If you add a replica administrative node after you deploy the cluster, you must update the configuration for each engine node.

    1. Complete steps 1-11 of Configuring replica administrative nodes. Make the following adjustments to step 2 and step 5:
        1. To complete step 2, the host you specify must be a resolvable DNS name for the node or the node's IP address. The port must be the TCP port that PingAccess listens to for the administrative interface. By default, this port is 9090.
        2. To complete step 5, select the key pair that you created for the CONFIG QUERY listener in step 2c of this topic as the **Replica Administrative Node Trusted Certificate**.

4. Configure the engine nodes in the cluster one at a time. For each engine node:
    1. Complete steps 1-10 of Configuring engine nodes.
    2. On the engine node, open the `conf/run.properties` file in a text editor and change the `pa.operational.mode` value to `CLUSTERED_ENGINE`.
    3. Complete step 11 of Configuring engine nodes.

        If you specified a proxy for the engine node, see the *What to do next* section also.

> **Note**
>
> Alternately, you can configure each engine node with an auto-registration file. For more information, see Configuring engine nodes using an auto-registration file.

Next steps

1. Go to **Settings → System → Clustering** to check your cluster's status. If everything is configured properly, the cluster engine nodes and the replica administrative node should display a green status icon, indicating that the cluster is operational. For more information about status icons, see Clustering in PingAccess.
2. Optionally, you can configure each node in the cluster to run PingAccess as a service. This set-up prompts PingAccess to run automatically when you start a node. For more information, see Running PingAccess as a service in *Installing and Uninstalling PingAccess*.

Promoting the replica administrative node

If the primary administrative node fails, promote the replica administrative node to the primary administrative node.

About this task

The replica administrative node is intended to be used for disaster recovery purposes. If you can recover the clustered console, then you should focus on recovery 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.

Promote the replica administrative node manually by making file system changes, or, in DevOps environments, use the replica admin API to temporarily promote the replica administrative node.

Manual promotion

Manually promoting the replica administrative node

About this task

To promote the replica admin manually:

Steps

1. Open the `<PA_HOME>/conf/run.properties` file in a text editor.
2. Locate the `pa.operational.mode` line and change the value from `CLUSTERED_CONSOLE_REPLICA` to `CLUSTERED_CONSOLE`.

   These properties are case-sensitive.

   > **Important**
   >
   > Do not restart the replica node during the promotion process. PingAccess can detect and apply this change without a restart, and restarting the node during its promotion can cause file corruption or failure to promote correctly.

Next steps

Complete Reinstating a replica administrative node after failing over.

Temporary admin API promotion

Using the admin API to temporarily promote the replica administrative node
About this task

In DevOps environments, use this method to promote the replica administrative node to a primary administrative node without requiring file system changes.

> **Important**
>
> This change only lasts until you restart the promoted node. To make this change permanent, you must complete the Manually promoting the replica administrative node procedure after restarting the promoted node.
> The temporary promotion method is only intended for DevOps environments.

> **Tip**
>
> You can use the **POST /adminConfig/replicaAdmin/status** endpoint to check the status of the replica admin. For example, you can review the node's identifier and check the last time its configuration was modified.

Steps

1. Sign on to the replica administrative PingAccess system and start a non-Internet Explorer (IE) browser.
2. Go to the API doc page at https://*<host>*:*<replica-admin-port>*/pa-admin-api/v3/api-docs/ and sign on.

   Example:

   https://localhost:9005/pa-admin-api/v3/api-docs/
3. Click and expand the **/adminConfig** section, then expand **POST /adminConfig/replicaAdmin/promote**.

   This endpoint is only available on the replica administrative server, and only if the primary server is unavailable.
4. Fill out the configuration as necessary:

Property	Description
`editRunPropertyFile`	If set to `true`, the admin API changes the value of the `pa.operational.mode` property from `CLUSTERED_CONSOLE_REPLICA` to `CLUSTERED_CONSOLE` in the `run.properties` file.
`editPrimaryHostPort`	If set to `true`, the admin API updates the primary administrative node's configuration with the promoted replica administrative node's **host:port** value.
`editPrimaryHttpProxy`	If set to `true`, the admin API updates the primary administrative node's configuration with the promoted replica administrative node's **HTTP Proxy** value.

Property	Description
`editPrimaryHttpsProxy`	If set to `true`, the admin API updates the primary administrative node's configuration with the promoted replica administrative node's **HTTPS Proxy** value.

5. Click **Try it out**.

Next steps

> **Note**
> This change only lasts until you restart the promoted node.

> **Important**
> If you want the promoted replica node to remain the primary administrative node, complete the Manually promoting the replica administrative node procedure after restarting the promoted node. Then complete Reinstating a replica administrative node after failing over.
> Otherwise, the promoted replica node will return to being the replica administrative node.

Reinstating a replica administrative node after failing over

If you fail over to your replica administrative node, you must configure a new replica admin node.

About this task

> **Note**
> If you want to then switch back to the original admin console, you must recreate it as a replica administrative node, then fail over to it.

Steps

1. Install the new replica administrative node.
2. In the `run.properties` file, change the value for `pa.operational.mode` to `CLUSTERED_CONSOLE_REPLICA`.

   This property is case-sensitive.
3. In the PingAccess admin console, edit the primary and replica host configuration and download the new replica node's bootstrap file:
   1. Click **Settings** and go to **Clustering → Administrative Nodes**.
   2. Change the **Primary Administrative Node** host name and port to the failed-over node.
   3. Remove the **Replica Administrative Node** public key, then change the **Replica Administrative Node** host name and port to point to the new replica node.

> **Tip**
>
> If your key pair doesn't include a wildcard, you can use the same host name as the original console to avoid having to recreate the console key pair and the `bootstrap.properties` files for each engine.

4. Click **Save & Download** to download the bootstrap file for the replica administrative node.

4. Copy the downloaded file to the new replica administrative node's `<PA_HOME>/conf` directory and rename it to `bootstrap.properties`.
5. Edit the `<PA_HOME>/conf/run.properties` file on the new replica administrative node and change the value of `pa.operational.mode` to `CLUSTERED_CONSOLE_REPLICA`.
6. Start the new replica node.
7. To verify that replication is complete, look for the message `Configuration successfully synchronized with administrative node` in the `<PA_HOME>/log/pingaccess.log` file.

Configuration file reference

You can configure any of the following properties used by PingAccess at runtime in the `<PA_HOME>/conf/run.properties` file.

In a clustered environment, each node has a unique `run.properties` file. Because changes to the `run.properties` file can significantly impact performance, use an identical `run.properties` configuration on all engine nodes.

> **Note**
>
> Changes made to the `run.properties` file only take effect after you restart the PingAccess service on the given node.

> **Note**
>
> When storing passwords in `run.properties`, obfuscate them using the `obfuscate.bat or obfuscate.sh` utility to mask the password value. You can find this utility in the `<PA_HOME>/bin` folder.

If you're running PingAccess in FIPS mode, PingAccess ignores all Secure Sockets Layer (SSL) cipher and protocol settings in the `run.properties` file. Learn more about the protocols and ciphers used in this mode in Managing Federal Information Processing Standards (FIPS) mode.

Operational mode

pa.operational.mode

Controls the operational mode of the PingAccess server in a cluster. The following table describes the acceptable values:

Value	Description
`STANDALONE`	Use this value for a standalone (unclustered) PingAccess instance that runs both the administrative console and the engine. This is the default value.

Value	Description
`CLUSTERED_CONSOLE`	Use this value for the server instance that you want to use as the administrative console server. **Note** Only one engine in a cluster can run the administrative console.
`CLUSTERED_CONSOLE_REPLICA`	Use this value for the server instance that 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 administrative properties depending on what operational mode an engine is using:

- Define all engine and administrative properties when `pa.operational.mode` is set to `STANDALONE`.
- Define only administrative properties when using `CLUSTERED_CONSOLE` or `CLUSTERED_CONSOLE_REPLICA` mode.
- Define only engine properties when using `CLUSTERED_ENGINE` mode.

- Learn more about configuring nodes using clustered operational mode in the Clustering Reference Guide.
- Learn more about installingPingAccess in standalone mode in Installing and Uninstalling PingAccess.

Administrative properties

admin.port

Defines the TCP port on which the PingAccess administrative console runs. The default value is `9000`.

admin.bindAddress

Defines the Internet Protocol (IP) address that `admin.port` binds 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 the server's IP addresses.

admin.ssl.protocols

Defines the protocols for use with administrative HTTPS ports. The default value is `$\{tls.default.protocols}`, which uses the protocols specified by the `tls.default.protocols` property.

admin.ssl.ciphers

Defines the type of cryptographic ciphers available for use with administrative HTTPS ports. The default value is `$\{tls.default.cipherSuites}`, which uses the ciphers specified by the `tls.default.cipherSuites` property.

admin.acceptors

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

admin.backlog

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

admin.httptransport.coreThreadPoolSize

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

admin.httptransport.ioThreads

Defines the number of I/O threads for the admin host. The default value is `0`, which indicates that PingAccess should automatically calculate the appropriate number of I/O threads for the host.

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.auth

Overrides the administrator authentication method. For example, if single sign-on (SSO) authentication is enabled and becomes misconfigured, this property can be used to bypass the database configuration and force the use of Basic authentication. The default value is `default`. A value of `native` overrides the administrator authentication method, meaning that only the local administrator credentials can be used to access the PingAccess console.

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.max.request.bodylength

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

admin.ui.max.sessions

Defines the maximum number of sessions for the admin UI when admin single logout (SLO) is not enabled. The default value is `100`.

admin.export.encryption.mode

Specifies how sensitive data should be encrypted on export. The default value is `MASTER_KEY`, which uses the system default master key for encryption. The `PORTABLE_INSECURE` value uses a randomly generated key for each export and includes the key in the export data. This method allows the exported data to be imported anywhere, including another cluster with a different master key, but since it includes the key it can present a significant security risk.

admin.startup.config.import.failfast

Defines the behavior when attempting to import a configuration file on startup. A value of `true` stops at the first failure, while a value of `false` continues and notes all errors. The default value is `false`.

- Learn more about how some properties are configured during installation in Installing and Uninstalling PingAccess.
- Learn more about how some properties impact administrative use in PingAccess User Interface Reference Guide.
- Learn more about how some SSL properties are overridden in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.

Token provider communication settings

pa.default.availability.ondemand.maxRetries

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

pa.default.availability.ondemand.connectTimeout

Defines, in milliseconds, the amount of time to wait before trying to connect to the remote host. The default value is `1000 0`.

pa.default.availability.ondemand.retryDelay

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

pa.default.availability.ondemand.failedRetryTimeout

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

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 value is `-1`, which indicates no timeout.

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 value is `-1`, which indicates no timeout.

Learn more about token providers in the token providers section of the PingAccess User Interface Reference Guide.

Cluster configuration settings

clusterconfig.enabled

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

> **Note**
>
> This property is set to `false` by the PingAccess upgrade utility after a PingAccess cluster is upgraded from a version earlier than 4.0.

clusterconfig.port

Defines the optional port used for cluster configuration. The default value is `9090`.

clusterconfig.bindAddress

Defines the optional address used for cluster configuration. The default value is `0.0.0.0`.

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.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.httptransport.socketTimeout

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

clusterconfig.httptransport.ioThreads

Defines the number of I/O threads for the cluster configuration host. The default value is `0`, which indicates that PingAccess should automatically calculate the appropriate number of I/O threads for the host.

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.maxThreadPoolSize

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

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.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 value is `500`.

engine.polling.delay

Defines, in milliseconds, how long after the prior query to the administrative console that the engine begins a new query for configuration information. The default value is `2000`.

engine.polling.test.delay

Defines, in milliseconds, how long after detecting an engine's `lastUpdated` value to be `null` or `zero` that PingAccess should wait before double-checking the value. The default value is `6000`.

> **Note**
>
> This property determines whether a replacement engine should self-register if PingAccess detects that there's an existing engine with the same name that isn't running. Changing this value changes PingAccess's behavior according to the following table:

engine.polling.test.delayvalue	PingAccess behavior
A negative number	Self-registration always fails, even if the existing engine with the same name was never updated or isn't polling.
A number from `0` up to the `engine.polling.delay` value	Self-registration happens automatically if the existing engine's `lastUpdated` value is `null` or `0`.
A number that's greater than the `engine.polling.delay` value	Self-registration happens if the existing engine's `lastUpdated` value is `null` or `0`. If the value is greater than `0`, PingAccess waits for a number of milliseconds equal to the `engine.polling.test.delay` value, then checks the `lastUpdated` value a second time. If the value doesn't change, PingAccess allows self-registration. Otherwise, this indicates that the existing engine is active, so PingAccess fails self-registration for the replacement engine.

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 value is `500`.

admin.polling.delay

Defines, in milliseconds, how long after the prior query to the administrative console that the replica administrative node begin a new query for configuration information. The default value is `2000`.

pa.config.replication.readTimeout

Defines, in milliseconds, the amount of time to wait before timing out the read response for the administrative node. The default value is `30000`.

pa.config.replication.maxRetries

Defines the maximum number of retries before marking the administrative node system down. The default value is `5`.

pa.config.replication.connectTimeout

Defines, in milliseconds, the amount of time to wait before trying to connect to the administrative node. The default value is `5000`.

pa.config.replication.retryDelay

Defines, in milliseconds, the amount of time to wait after a timeout before retrying the administrative node. The default value is `2000`.

pa.config.replication.failedRetryTimeout

Defines, in seconds, the amount of time to wait before retrying a failed connection to the administrative node. The default value is `-1`, which indicates no timeout.

pa.config.replication.pooledConnectionTimeout

Defines, in milliseconds, the amount of time to wait before timing out the request for a pooled connection to the administrative node. The default value is `-1`, which indicates no timeout.

Learn more about cluster configuration in the Clustering Reference Guide.

Engine properties

engine.http.bindAddress

Defines the address for an engine in a clustered environment. The default value is `0.0.0.0`.

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.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.http.enabled

Defines whether the engine node (either `STANDALONE` or `CLUSTERED_ENGINE`, if you're running PingAccess in a cluster) listens for requests on the ports defined by the engine listeners. 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.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.httptransport.ioThreads

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

engine.websocket.maxConnections

Sets the maximum number of allowed web socket connections. The default value is `-1`, which denotes no limit.

engine.ssl.protocols

Defines the protocols used with engine HTTPS ports. The default value is `TLSv1, TLSv1.1, TLSv1.2, TLSv1.3`.

engine.ssl.ciphers

Defines the type of cryptographic ciphers available for use with engine HTTPS ports. The default value is `$\{tls.default.cipherSuites}`, which uses the ciphers specified by the `tls.default.cipherSuites` property.

client.ioThreads

Defines the number of threads for client connections to backend sites. The default value is `0`, which denotes no limit.

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.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.limitRequestLine

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

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.maxConnectionsPerSite

Defines the maximum number of connections PingAccess will open to the PingFederate admin or engine. The default value is `-1`, which denotes no limit.

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.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 PingAccess-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].$,\
^.*UCBrowser\\/12.13.[0-2][\\.\\d].*$
```

pa.default.session.cookie.attributes.partitioned

When enabled, adds the `Partitioned` attribute to cookies set by PingAccess. This ensures that cross-site cookies will continue to be readable within the same context that they're created in. Learn more in the PingAccess 8.1 (June 2024) release notes.

The default value is `false`. If you edit this value, restart PingAccess to make your changes take effect.

pa.default.cookie.attributes.partitioned.excludedUserAgentPatterns

If `pa.default.session.cookie.attributes.partitioned` is enabled, or if you've selected **Partitioned Cookie** on a web session or the admin web session, you can define a comma-separated list of regex to declare any user-agents that don't support the `Partitioned` attribute. If the user-agent header from a request matches any of the values in the list, PingAccess excludes the `Partitioned` attribute from any related cookies that it sets.

For example:

```
pa.default.cookie.attributes.partitioned.excludedUserAgentPatterns= ^.\\(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].$,\

^.\\(Macintosh;.Mac OS X 10_14[\\d].\\) AppleWebKit\\/[\\.\\d]+ \\(KHTML. like Gecko\\)$,\

Box.\\/.+Darwin\\/10.14.$,\

^.*PAN GlobalProtect.*Mac OS X 10.*14.$
```

By default, this property doesn't have a value. If you edit this value, restart PingAccess to make your changes take effect.

pa.uri.strict

When enabled, this setting requires that the raw input Uniform Resource Identifier (URI) be in strict compliance with the URI spec implemented by `java.net.URI` when generating URIs. The default value is `false`.

pa.uri.canonicalize

When enabled, PingAccess normalizes empty and dot path segments that contain URL-encoded forward slashes (`/` , encoded as `%2f`) or periods (encoded as `%2e`). When this setting has a value of `false` , PingAccess doesn't normalize empty and dot path segments that contain URL-encoded forward slashes or periods. The default value is `true` .

- Learn more about cluster configuration in the Clustering Reference Guide.
- Learn more about how some SSL properties are overridden in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.

Agent properties

agent.http.port

Defines the TCP port on which the engine listens for agent requests. The default value is `3030` .

agent.http.bindAddress

Defines the address from which an engine listens for agent requests. The default value is `0.0.0.0` .

agent.http.acceptors

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

agent.http.secure

Defines whether the engine is using HTTPS for agent requests. The default value is `true`.

agent.http.backlog

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

agent.http.enabled

Defines whether the engine node (either `STANDALONE` or `CLUSTERED_ENGINE`, if you're running PingAccess in a cluster) listens for agent requests on the port defined by the `agent.http.port` setting. The default value is `true`.

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.ssl.protocols

Defines the protocols used for communication with agent HTTPS ports. The default value is `${tls.default.protocols}`, which uses the protocols specified by the `tls.default.protocols` property.

agent.ssl.ciphers

Defines the type of cryptographic ciphers available for use with agent HTTPS ports. The default value is `${tls.default.cipherSuites}`, which uses the ciphers specified by the `tls.default.cipherSuites` property.

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.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.httptransport.ioThreads

Defines the number of I/O threads for the agent host. The default value is `0`, which denotes that PingAccess should automatically calculate the appropriate number of I/O threads for the host.

agent.authz.header.required

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

agent.default.token.cache.ttl

Defines, in seconds, the time to live for cached agent tokens. The default value is `60`.

- Learn more about agent settings in PingAccess User Interface Reference Guide.

- Learn more about agent installation and management for Apache (RHEL) in PingAccess Agent for Apache (RHEL).
- Learn more about agent installation and management for Apache (SLES) in PingAccess Agent for Apache (SLES).
- Learn more about agent installation and management for Apache (Windows) in PingAccess Agent for Apache (Windows).
- Learn more about agent installation and management for IIS in PingAccess Agent for IIS.
- Learn more about agent installation and management foe NGINX in PingAccess Agent for NGINX.
- Learn more about how some SSL properties are overridden in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.

Sideband properties

sideband.http.port

Defines the TCP port on which the engine listens for sideband requests. The default value is `3020`.

sideband.http.bindAddress

Defines the address from which an engine listens for sideband requests. The default value is `0.0.0.0`.

sideband.http.acceptors

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

sideband.http.secure

Defines whether the engine is using HTTPS for sideband requests. The default value is `true`.

sideband.http.backlog

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

sideband.http.enabled

Defines whether the engine node (either `STANDALONE` or `CLUSTERED_ENGINE`, if you're running PingAccess in a cluster) listens for sideband requests on the port defined by the `sideband.http.port` setting. The default value is `false`.

sideband.http.reuseAddress

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

sideband.ssl.protocols

Defines the protocols used for communication with sideband HTTPS ports. The default value is `${tls.default.protocols}`, which uses the protocols specified by the `tls.default.protocols` property.

sideband.ssl.ciphers

Defines the type of cryptographic ciphers available for use with sideband HTTPS ports. The default value is `${tls.default.cipherSuites}`, which uses the ciphers specified by the `tls.default.cipherSuites` property.

sideband.httptransport.coreThreadPoolSize

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

sideband.httptransport.maxThreadPoolSize

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

sideband.httptransport.socketTimeout

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

sideband.httptransport.ioThreads

Defines the number of I/O threads for the sideband host. The default value is `0`, which denotes that PingAccess should automatically calculate the appropriate number of I/O threads for the host.

- Learn more about how to configure a sideband client in the user interface in Sideband Clients.
- Learn more about how some SSL properties are overwritten in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.

URL filtering settings

pa.interceptors.relativepath.strict

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

pa.interceptors.relativepath.decode.count

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

pa.interceptors.relativepath.decode.regex

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.
> Learn more about URL filtering in Adding rewrite URL rules.

Monitoring

pa.mbean.site.connection.pool.enable

When set to `true`, enables Java Management Extensions (JMX) read-only access to backend connection pools. This can be useful when troubleshooting latency issues because it provides information about requests that are waiting for a connection to targets in a site when `maxConnections` is not unlimited. The default value is `false`.

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`.

pa.statistics.window.seconds

If the `enable.detailed.heartbeat.response` property is set to `true`, this property sets the number of seconds back to collect response statistics. A value less than `1` disables collection. The default value is `0`.

Learn more about monitoring in the PingAccess Monitoring Guide.

TLS/SSL

tls.default.protocols

Defines the default protocols used for HTTPS communication. The default value is `TLSv1.1, TLSv1.2, TLSv1.3`.

tls.default.cipherSuites

Defines the default set of ciphers used for HTTPS communication. The default value is:

```
TLS_CHACHA20_POLY1305_SHA256,\
TLS_AES_256_GCM_SHA384,\
TLS_AES_128_GCM_SHA256,\
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
```

Note

Legacy browsers might 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`

clusterconfig.ssl.protocols

Defines the protocols used for communication with HTTPS ports in a clustered configuration. The default value is `$\{tls.default.protocols}`, which uses the protocols specified by the `tls.default.protocols` property.

clusterconfig.ssl.ciphers

Defines the type of cryptographic ciphers available for use with HTTPS ports in a clustered configuration. The default value is `$\{tls.default.cipherSuites}`, which uses the ciphers specified by the `tls.default.cipherSuites` property.

site.ssl.protocols

Defines the protocols used for communication with site HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the Java Development Kit (JDK).

site.ssl.ciphers

Defines the type of cryptographic ciphers available for use with site HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

pf.ssl.protocols

Defines the protocols used for communication with PingFederate HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

pf.ssl.ciphers

Defines the type of cryptographic ciphers available for use with PingFederate HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

provider.ssl.protocols

Defines the protocols used for communication with provider HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

provider.ssl.ciphers

Defines the type of cryptographic ciphers available for use with provider HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

as.ssl.protocols

Defines the protocols used for communication with authorization server HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

as.ssl.ciphers

Defines the type of cryptographic ciphers available for use with authorization server HTTPS ports. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

p14c.ssl.protocols

Defines the protocols used for communication with PingOne. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

p14c.ssl.ciphers

Defines the type of cryptographic ciphers available for use with PingOne. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

thirdpartyservice.ssl.protocols

Defines the protocols used for communication with third-party services. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

thirdpartyservice.ssl.ciphers

Defines the type of cryptographic ciphers available for use with third-party services. There is no default value. When not specified, PingAccess uses the protocols defined in the JDK.

- Learn more about the use of TLS/SSL settings for security in the PingAccess Hardening Guide.
- Learn more about how some TLS properties are overwritten in FIPS mode in Managing Federal Information Processing Standards (FIPS) mode.

POST preservation properties

pa.oidc.post.preservation.encrypt

When enabled, PingAccess preserves POST data 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. The default value is `postParams`.

- Learn more about the use of POST preservation in system templates meant to provide information to the end user in User-facing page customization reference.
- Learn more about the use of POST preservation in system templates meant to provide localized versions of user-facing status messages generated by PingAccess in User-facing page localization reference.
- Learn more about the use of POST preservation in web sessions in the PingAccess User Interface Reference Guide.

Configuration database and key store settings

derby.language.statementCacheSize

Defines the number of statements that are stored in memory. The default value is `500`.

derby.storage.pageCacheSize

Defines the number of pages cached in memory. The default value is `1000`.

pa.trust.keystore.type

Defines the truststore type for the `$JAVA_HOME/lib/security/cacerts` keystore. The default value is `JKS`.

> **Note**
>
> PKCS#12 isn't supported if you're using FIPS mode, because it isn't FIPS-compliant.

pa.trust.keystore.path

Defines the path for the `$JAVA_HOME/lib/security/cacerts` keystore. The default value is `${java.home}/lib/security/cacerts`.

pa.keystore.pw

Defines the password for the truststore configured in the `pa.trust.keystore.path` property. The value is encrypted.

- Learn more about the initial database settings in Installing and Uninstalling PingAccess.
- Learn more about adjusting PingAccess settings for specific environments in the Performance Tuning Reference Guide.
- Learn more about how the native Apache Derby properties are used in the Apache Derby: Documentation in the Apache documentation.

PingFederate administration integration properties

pf.api.maxRetries

Defines the maximum number of retries PingAccess attempts to make to the PingFederate server before declaring the server unavailable. The default value is `0`.

pf.api.socketTimeout

Defines, in milliseconds, the socket timeout for the PingFederate API endpoint. The default value is `5000`.

pf.api.maxConnections

Defines the maximum number of connections PingAccess will establish to the PingFederate API endpoint. The default value is `-1`, which means there is no limit.

pf.api.keepAliveTimeout

Defines, in milliseconds, the keep alive timeout for the PingFederate API. The default value is `30000`.

pf.api.readTimeout

Defines, in milliseconds, how long the API will wait for responses from PingFederate when making calls to the PingFederate administrative API. The default value is `-1`, which means there is no limit.

Learn more about using PingAccess with PingFederate in:

- Configure PingFederate as the token provider for PingAccess.
- PingAccess User Interface Reference Guide.

Administrative console settings

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 sign-on process in large environments. If you disable the creation of backup files, use the administrative API backup endpoint to create regular backups.

pa.admin.user.password.regex

Defines the regex that controls password complexity for the administrative console. The default value is:

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

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.test.connections

A boolean property that allows the PingAccess administrative console to make HTTP calls to validate that it can reach PingFederate and sites when the user configures them. The default value is `true`.

account.locking.max.consecutive.failures

Defines the maximum number of failed sign-on attempts before locking the account when using basic authentication in the administrative console 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`.

Learn more about PingAccess administration in the PingAccess User Interface Reference Guide.

EHCache configuration properties

pa.ehcache.PingFederateReferenceTokenCache.maxEntriesLocalHeap

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

pa.ehcache.PingFederateReferenceTokenCache.timeToIdleSeconds

Defines, in seconds, the time an entry in the OAuth token cache can be idle before it is expired. The default value is `0`.

pa.ehcache.PingFederateReferenceTokenCache.timeToLiveSeconds

Defines, in seconds, the maximum time an entry can be in the OAuth token cache. The default value is `0`.

pa.ehcache.ServiceTokenCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for token mediation. The default value 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 value is `1800`.

pa.ehcache.ServiceTokenCache.timeToLiveSeconds

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

pa.ehcache.PATokenValidationCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for decryption of signed or encrypted PingAccess tokens. The default value 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 value is `120`.

pa.ehcache.PATokenValidationCache.timeToLiveSeconds

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

pa.ehcache.PFSessionValidationCache.maxEntriesLocalHeap

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

pa.ehcache.PFSessionValidationCache.timeToIdleSeconds

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

pa.ehcache.PFSessionValidationCache.timeToLiveSeconds

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

pa.ehcache.PAWamUserAttributesCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for the PingAccess Web Access Management (WAM) user attribute cache. The default value is `10000`.

pa.ehcache.PAWamUserAttributesCache.timeToIdleSeconds

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

pa.ehcache.PAWamUserAttributesCache.timeToLiveSeconds

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

pa.ehcache.AuthTokenCache.maxEntriesLocalHeap

Defines the maximum size of the JSON Web Token (JWT) identity mapping token cache used when sending tokens to a protected site. The default value is `10000`.

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. The default value is `10000`.

pa.ehcache.AzureGroupNameCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for the Azure group name cache. The default value is `10000`.

Learn more about EHCache configuration in the Clustering Reference Guide.

Security headers properties

admin.headers

Additional headers added to responses from the PingAccess administrative console and the administrative API interface. Define header values using the `admin.header` prefix. The default value is:

```
X-Frame-Options,X-XSS-Protection,X-Content-Type-Options,Strict-Transport-Security,Content-Security-Policy
```

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 administrative console. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

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 administrative console. The default value is `1; mode=block`.

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 administrative console. The default value is `nosniff`.

admin.header.Content-Security-Policy

Sets the parameters for the `content-security-policy` response header sent by PingAccess in response to API calls. The default value is:

```
default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; font-src 'self' data:;
```

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 administrative console. This property is commented out by default and should be enabled only if the admin and engine use different host names. The default value is `max-age=31536000; includeSubDomains`.

agent.assets.headers

Additional headers added to responses from PingAccess agents. Header values are defined using the `agent.assets.header` prefix. The default value is `X-Frame-Options`.

agent.assets.header.X-Frame-Options

Sets the parameters for the `X-Frame-Options` HTTP response header sent to the browser using the agent when responding to a request for an asset used by a PingAccess template. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

agent.error.headers

Additional headers added to error responses from PingAccess agents. Header values are defined using the `agent.error.header` prefix. The default value is `X-Frame-Options, Content-Security-Policy`.

> **Note**
>
> `Content-Security-Policy` might be omitted if PingAccess was upgraded with template customizations. To enable for this case, add the `Content-Security-Policy` value to this property and uncomment `agent.error.header.Content-Security-Policy`.

agent.error.header.Content-Security-Policy

Sets the parameters for the `Content-Security-Policy` HTTP response header sent to the browser using the agent when responding with a PingAccess error template. The default value is:

```
default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; font-src 'self' data:; object-src 'none';
```

> **Note**
>
> This property might be commented out if PingAccess was upgraded with template customizations.

agent.error.header.X-Frame-Options

Sets the parameters for the `X-Frame-Options` HTTP response header sent to the browser using the agent when responding with a PingAccess error template. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

engine.assets.headers

Additional headers added to responses from the PingAccess engine. Header values are defined using the `engine.assets.header` prefix. The default value is `X-Frame-Options`.

engine.assets.header.X-Frame-Options

Sets the parameters for the `X-Frame-Options` HTTP response header sent to the browser using the engine when responding to a request for an asset used by a PingAccess template. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

engine.error.headers

Additional headers added to error responses from the PingAccess engine. Define header values using the `engine.error.header` prefix. The default value is `X-Frame-Options, Content-Security-Policy`.

> **Note**
>
> `Content-Security-Policy` might be omitted if PingAccess was upgraded with template customizations. If you want to enable for this case, add the `Content-Security-Policy` value to this property and uncomment `engine.error.header.Content-Security-Policy`.

engine.error.header.Content-Security-Policy

Sets the parameters for the `Content-Security-Policy` HTTP response header sent to the browser using the engine when responding with a PingAccess error template. The default value is:

```
default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; font-src 'self' data:; object-src 'none';
```

> **Note**
>
> This property might be commented out if PingAccess was upgraded with template customizations.

engine.error.header.X-Frame-Options

Sets the parameters for the `X-Frame-Options` HTTP response header sent to the browser using the engine when responding with a PingAccess error template. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

sideband.assets.headers

Additional headers added to responses from PingAccess sideband clients. Define header values using the `sideband.assets.header` prefix. The default value is `X-Frame-Options`.

sideband.assets.header.X-Frame-Options

Sets the parameters for the `X-Frame-Options` HTTP response header sent to the browser using the sideband client when responding to a request for an asset used by a PingAccess template. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

> **Note**
>
> `Content-Security-Policy` might be omitted if PingAccess was upgraded with template customizations. To enable for this case, add the `Content-Security-Policy` value to this property and uncomment `sideband.error.header.Content-Security-Policy`.

sideband.error.header.Content-Security-Policy

Sets the parameters for the `Content-Security-Policy` HTTP response header sent to the browser using the sideband client when responding with a PingAccess error template. The default value is:

```
default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; font-src
'self' data:; object-src 'none';
```

> **Note**
>
> This property might be commented out if PingAccess was upgraded with template customizations.

sideband.error.headers

Additional headers added to error responses from PingAccess sideband clients. Define header values using the `sideband.error.header` prefix. The default value is `X-Frame-Options,Content-Security-Policy`.

> **Note**
>
> `Content-Security-Policy` might be omitted if PingAccess was upgraded with template customizations. To enable for this case, add the `Content-Security-Policy` value to this property and uncomment `sideband.error.header.Content-Security-Policy`.

sideband.error.header.X-Frame-Options

Sets the parameters for the `X-Frame-Options` HTTP response header sent to the browser using the sideband client when responding with a PingAccess error template. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

pf.redirect.use.default.csp

Determines whether PingAccess uses the default CSP for HTML authentication challenge responses (ACRs) for web apps. The default value is `true`.

If you set the `pf.redirect.use.default.csp` property to `false`, you can use the `pf.redirect.headers` to set the CSP that PingAccess uses.

> **Note**
>
> Make sure to use the `pf.redirect.headers` to configure a value if you set `pf.redirect.use.default.csp` to `false`. Otherwise, PingAccess won't use any CSP.

pf.redirect.headers

Additional headers added to the redirection response that sends the client to PingFederate for authentication. These headers are added when using the SPA support disabled Authentication Challenge Policy, using the global PingFederate Redirect Headers Appender challenge response filter, or using an application that is configured without an Authentication Challenge Policy and SPA support disabled. Define header values using the `pf.redirect.header` prefix. The default value is `X-Frame-Options,Content-Security-Policy`.

> **Note**
>
> `Content-Security-Policy` might be omitted if PingAccess was upgraded with template customizations. If you want to enable for this case, add the `Content-Security-Policy` value to this property and uncomment `pf.redirect.header.Content-Security-Policy`.

pf.redirect.header.Content-Security-Policy

Sets the parameters for the `Content-Security-Policy` HTTP response header that is sent when the user is redirected to PingFederate to authenticate. The default value is:

```
default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; font-src
'self' data:; object-src 'none';
```

> **Note**
>
> This property might be commented out if PingAccess was upgraded with template customizations.

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. The default value is `DENY`.

Learn more about this header and its potential values in x-frame-options.

rule.error.headers

Additional headers added to responses that result from policy rule results. Define header values using the `rule.error.header` prefix. The default value is `Content-Security-Policy`.

> **Note**
>
> `Content-Security-Policy` might be omitted if PingAccess was upgraded with template customizations. To enable for this case, add the `Content-Security-Policy` value to this property and uncomment `rule.error.header.Content-Security-Policy`.

rule.error.header.Content-Security-Policy

Sets the parameters for the `Content-Security-Policy` HTTP response header sent to the browser when the response is generated by a rule failure. The default value is:

```
default-src 'self'; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; font-src
'self' data:; object-src 'none';
```

> **Note**
>
> This property might be commented out if PingAccess was upgraded with template customizations.

oauth.error.headers

Additional headers added to responses that result from requests made to a protected API application that lack a valid OAuth Bearer token. Define header values using the `oauth.error.header` prefix. The default value is `Content-Security-Policy`.

oauth.error.header.Content-Security-Policy

Sets the parameters for the `Content-Security-Policy` HTTP response header sent to the browser when PingAccess receives a request made to a protected API application that doesn't contain a valid OAuth Bearer token. The default value is:

```
default-src 'self'; style-src 'self'; \ script-src 'self'; font-src 'self' data:; object-src 'none';
```

You can find more information about security headers in:

- Learn more about security headers and the behavior of the administrative API in PingAccess API endpoints.
- Learn more about security headers and administrative console settings in the PingAccess User Interface Reference Guide.
- Learn more about security headers and measures to ensure security in the PingAccess Hardening Guide.

Localization settings

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. The default value is `true`.

pa.localization.missing.message.placeholder

Defines the message used when an error message is unresolvable. There is no default value.

- Learn more about localization and customizing PingAccess page templates and understanding the difference between customizable templates and system templates in User-facing page customization reference.
- Learn more about localizing user-facing system status messages in User-facing page localization reference.

PingAccess deployment guide

Use the deployment guide to decide how PingAccess fits into your existing network, such as determining the deployment architecture required for your use case and whether you require high-availability options.

This section provides information to help you make the right decisions for your environment in PingAccess.

Use cases and deployment architecture

Depending on your needs and infrastructure capabilities, there are many options for deploying PingAccess in your network environment.

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 can add headers and tokens required by the target resource. Under the PingAccess policy server's control, the agent might 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, such as 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 PingAccess token in a cookie derived from the user's profile during an OpenID Connect (OIDC) based sign on at PingFederate.

- Evaluates application and resource-level policies and validates the tokens in conjunction with an OIDC Policy configured within PingFederate

- Acquires the appropriate target security token (site authenticators) from the PingFederate Security Token Service (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 enterprise resources available online, 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 (OIDC) 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 expanding mobile device and application programming interface (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 Security Token Service (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 sideband API access management

A PingAccess API access management sideband deployment enables an organization to quickly set up an environment that provides a secure method of managing access rights to API-based applications while integrating with existing identity management infrastructure and minimal network configuration changes.

With growing numbers of internal and external users, and more enterprise resources available online, ensure that qualified users can access only those applications to which they have permission. An application programming interface (API) access environment provides authentication and policy-based access management while integrating with existing infrastructure.

The PingAccess sideband plugin is installed on the API gateway serving the protected API applications and configured to communicate with PingAccess server also deployed on the network. When the API gateway intercepts a client request to a protected API resource, it performs the following actions:

- Intercepts inbound requests to API applications
- Sends requests to the PingAccess sideband API endpoint, sending along relevant request information needed by policy server

- Receives 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
- Sends requests to the PingAccess sideband API endpoint, sending along relevant response information needed by the policy server.
- Applies modifications from the policy server and relays response

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 (OIDC) Policy configured within PingFederate
- Acquires the appropriate HTTP request header configuration from the associated identity mappings
- Sends a response with instructions on whether to allow the request and how to modify the client request headers

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 backend sites.

With growing numbers of internal and external users, you need to know which users are accessing applications, where and when they are accessing them, and ensuring that they are correctly accessing only the applications to which they have permission. A standardized auditing and proxying deployment provides a centrally-controlled model that enforces existing infrastructure and security policies, safeguarding an organization's assets.

At the perimeter of a protected network, between mobile, in-browser, or server-based client applications and backend sites, PingAccess performs the following actions:

- PingAccess receives inbound calls requesting access to protected backend sites.
- PingAccess audits the request and then makes authorized requests to the backend sites.
- PingAccess receives and processes responses and relays them to the clients.

The following sections describe sample proof of concept and production architectures for an auditing and proxying use case deployment:

- Audit and Proxy POC Deployment Architecture
- Audit and Proxy Production Deployment Architecture

Configuration by use case

Configuration steps vary depending on what type of deployment you are implementing.

For a detailed discussion of deployment considerations and best practices in designing your architecture, see the Deployment Guide. 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

After you complete the above configuration settings, the following steps are similar for all use cases:

- Configure sites and agents to define the target applications you want protected. Sites might need site authenticators to define the credentials the site expects for access control.
- Configure applications and resources to define the assets you want to allow clients to access.
- Create policies for the defined applications and resources to protect them.

Web Access Management Gateway deployment table

The following table describes the important configuration options for a Web Access Management (WAM) Gateway deployment.

For specific use case information, see Deploying for Gateway Web Access Management in the Deployment Guide.

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.
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.

Step	Description
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 table

This table describes the important configuration options for a Web Access Management (WAM) agent deployment.

For specific use case information, see Deploying for Agent Web Access Management.

Deploy PingAccess agent using the following steps:

1. Install PingAccess agent on web server. For more information, see instructions 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 using the download field in the **Shared Secrets** field.
3. Deploy the agent `bootstrap.properties` file to agents. For more information, see 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.
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.

Step	Description
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 table

This deployment table describes the important configuration options for deploying an API Gateway.

For specific use case information, see Deploying for Gateway API Access Management in the Deployment Guide.

Step	Description
Configure the connection to the PingFederate OAuth Authorization Server.	PingAccess uses this connection and credentials to validate incoming access tokens for securing application programming interface (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 table

This gateway deployment table describes the important configuration options for an auditing or proxying deployment.

For specific use case information, see Deploying for Auditing and Proxying.

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

PingAccess uses Web Access Management (WAM) capabilities to allow organizations to manage access rights to web-based resources.

With growing numbers of internal and external users, and more and more enterprise resources available online, ensure that qualified users can access only those resources to which they have permission.

WAM is a form of identity management that controls access to web resources, providing authentication and policy-based access management. After a user is authenticated, PingAccess applies application and resource-level policies to the request. After policy evaluation is passed, any required identity mediation between the backend site and the authenticated user is performed. The user is then granted access to the requested resource.

PingAccess provides two deployment architectures for WAM - 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 using 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

Deploy PingAccess using Agents, as a Gateway (or reverse proxy), or using a combination of both. Before choosing a deployment, 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 or app server processing and performance
- Works with existing security token types, such as creating third party Web Access Management (WAM) tokens

Cons:

- Requires networking changes
- Requires strategy for securing direct access to backend web or app servers (network routing or service level authentication)
- Depending on the application, might 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

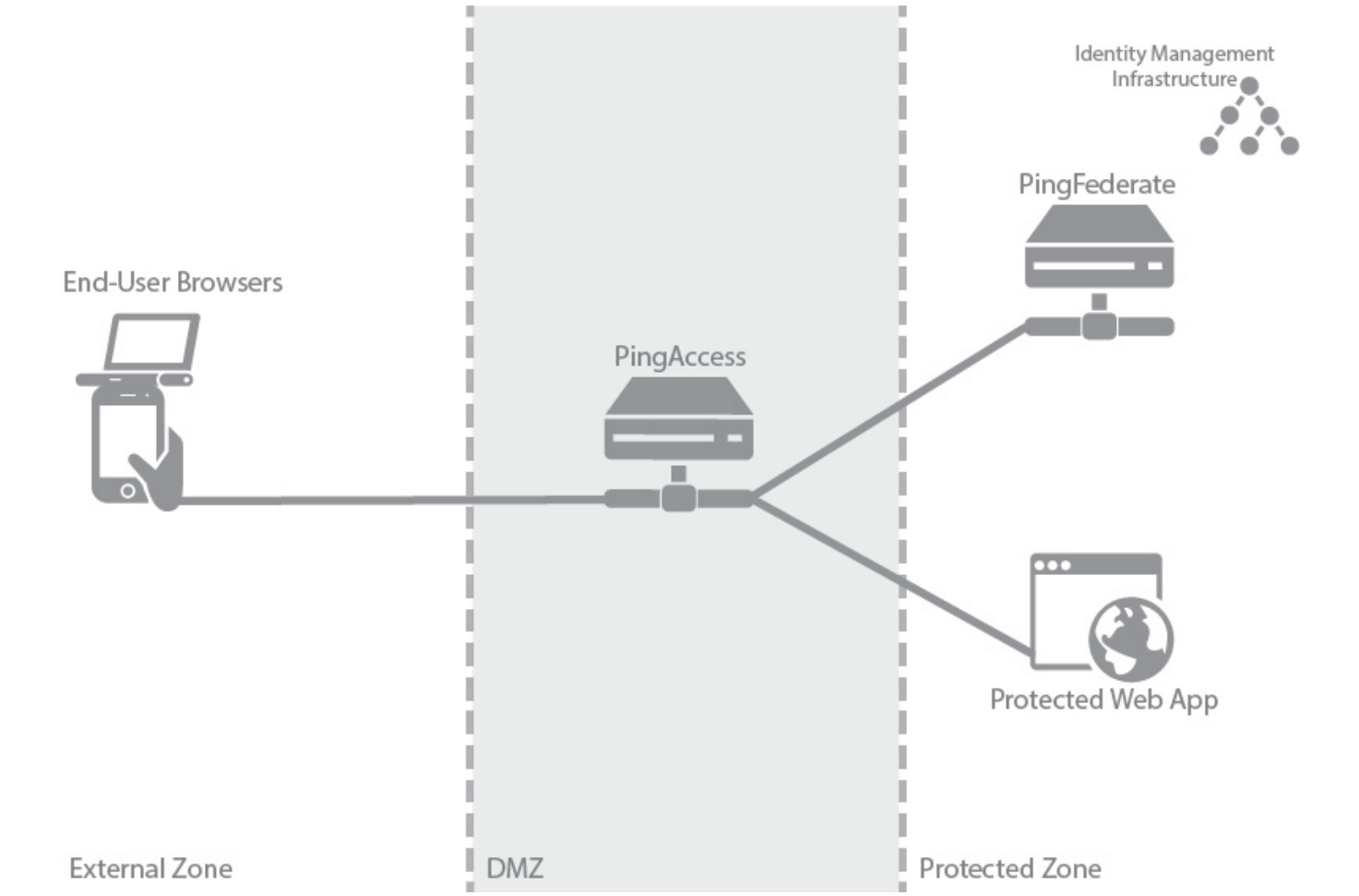
Cons:

- High cost of ownership when many agent instances are deployed, although should be upgradable or patchable independently of PingAccess policy server
- Policy evaluation is cached, and although periodically flushed or re-evaluated (for new sessions, updates to session token, etc.) , isn't as "real time" as proxy
- Tight dependency on web server version and platform

Web Access Management Gateway proof of concept deployment architecture

This proof of concept deployment environment is used to emulate a Web Access Management (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.

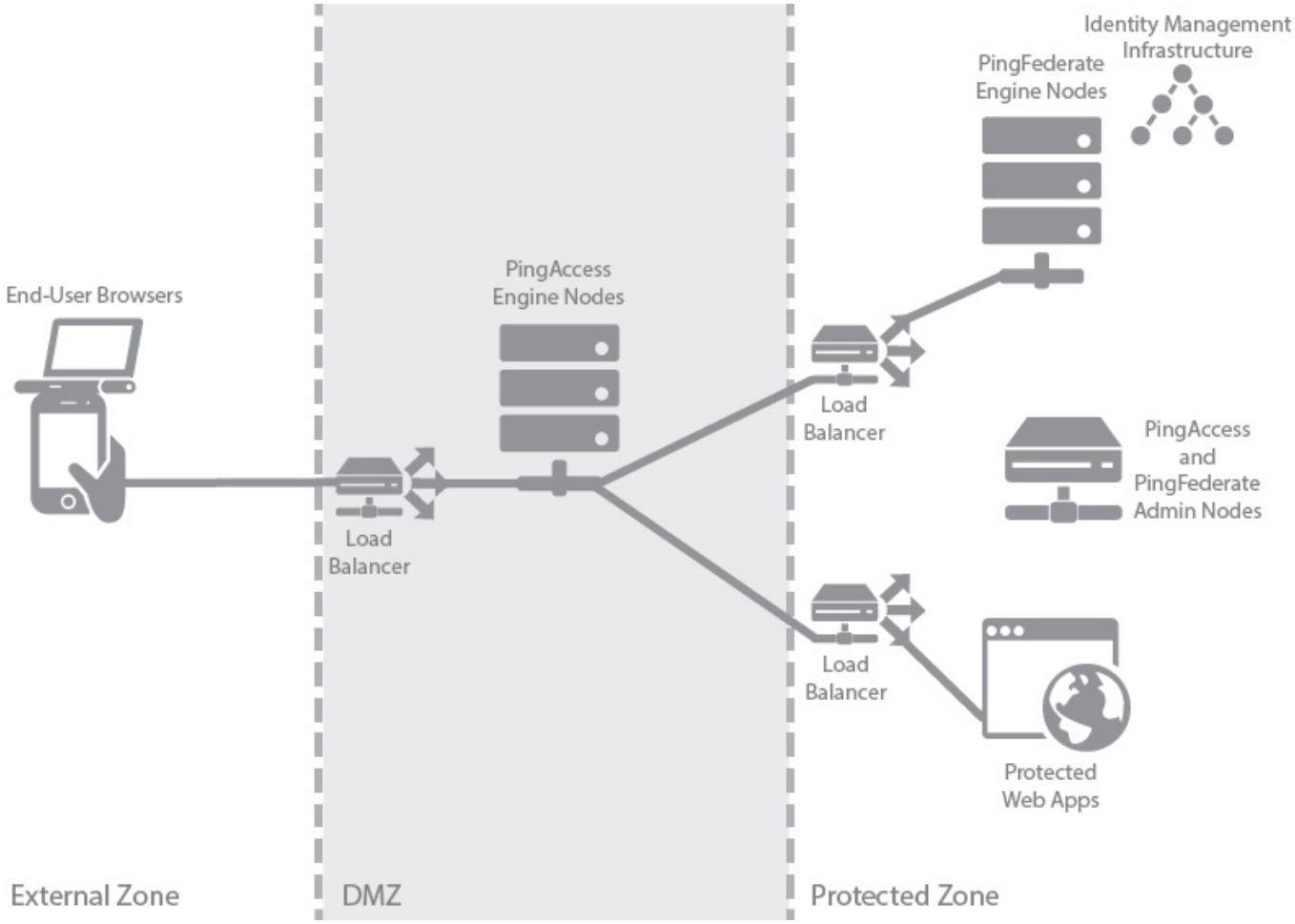
Zone	Description
External Zone	External network where incoming requests for web applications originate.
DMZ	Externally exposing segment where PingAccess is accessible to web browsers. PingAccess is a standalone instance in this environment, serving as both a runtime and an administrative port.
Protected Zone	Backend 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 to authenticate users, depicted by the icon in the diagram.

Web Access Management Gateway production deployment architecture

This environment shows a Web Access Management (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.

> **Note**
>
> PingAccess provides high availability and basic load balancing for the protected web apps in the protected zone. For more information, see the availability profiles and load balancing strategies documentation.



The following table describes the three zones within this proposed architecture.

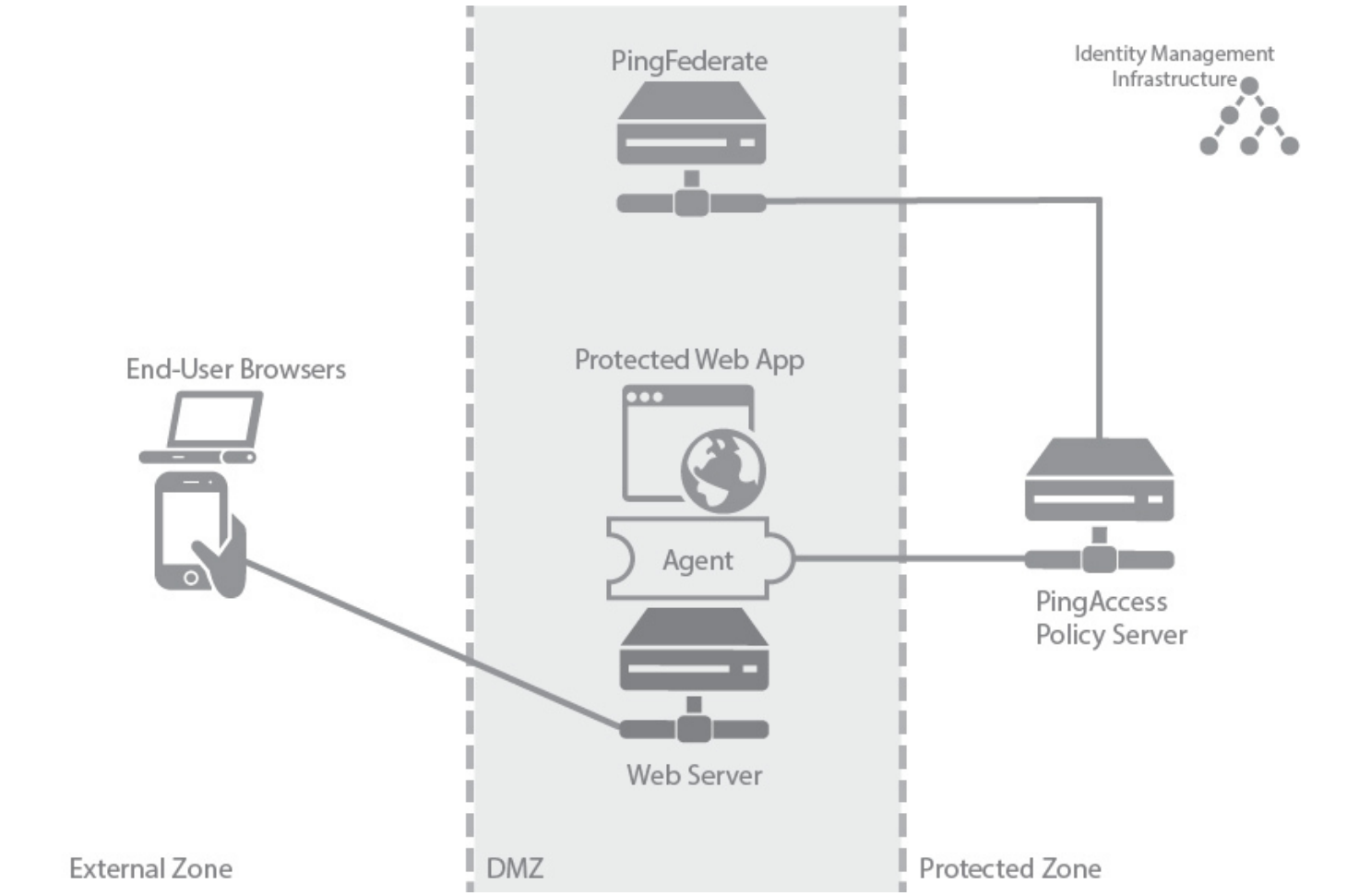
Zone	Description
External Zone	External network where incoming requests for web applications originate.

Zone	Description
DMZ	Externally exposing segment where PingAccess is accessible to web browsers. A minimum of two PingAccess engine nodes will be deployed in the DMZ to achieve high availability. Depending on your scalability requirements, more nodes might be required.
Protected Zone	Backend 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 can be co-located on a single machine to reduce hardware requirements.

Web Access Management Agent proof of concept deployment architecture

This proof of concept deployment environment emulates a Web Access Management (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.



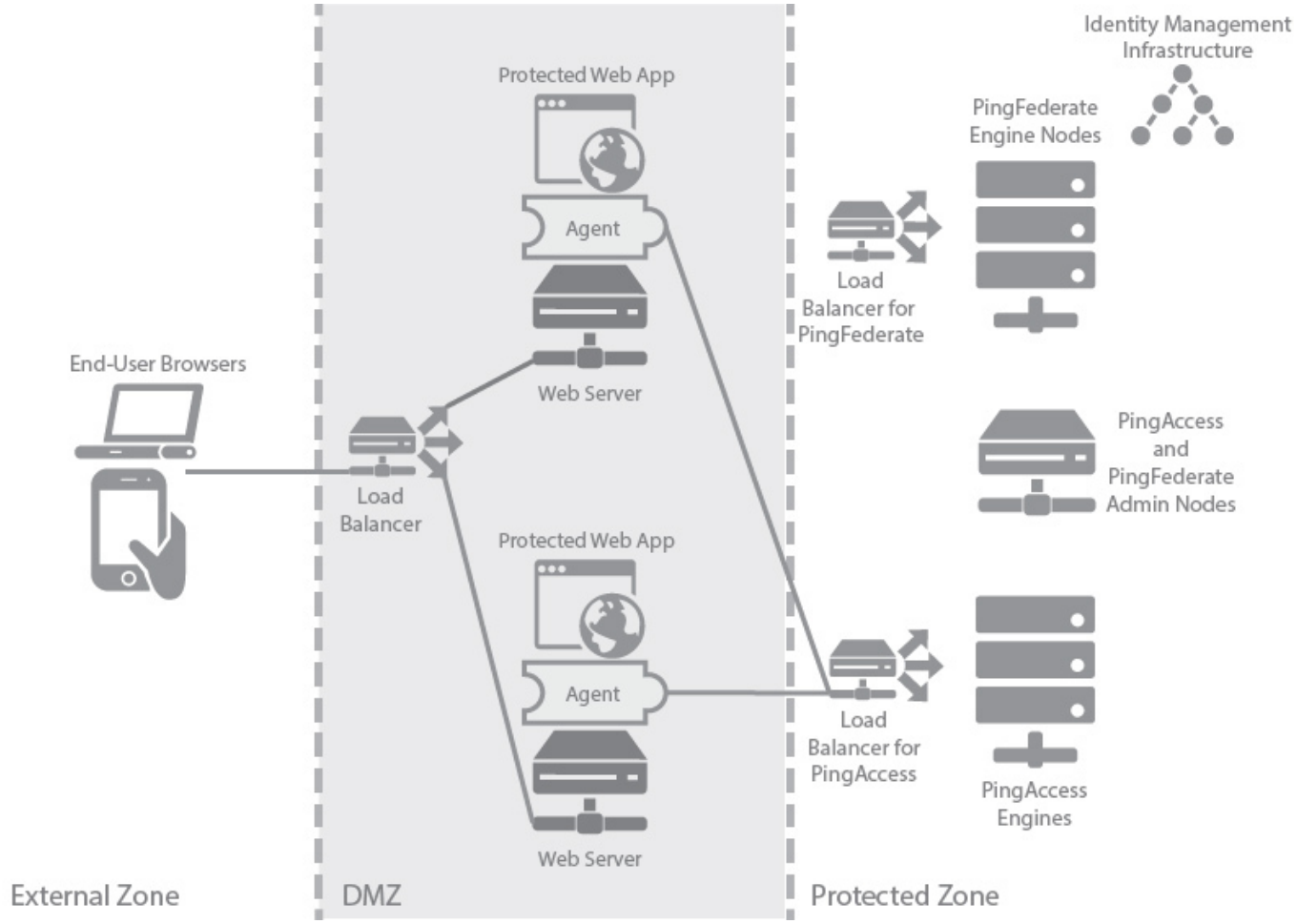
The following table describes the three zones within this proposed architecture.

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 to authenticate users.
Protected Zone	Backend 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

The deployment environment shows a Web Access Management (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.



The following table describes the three zones within this proposed architecture.

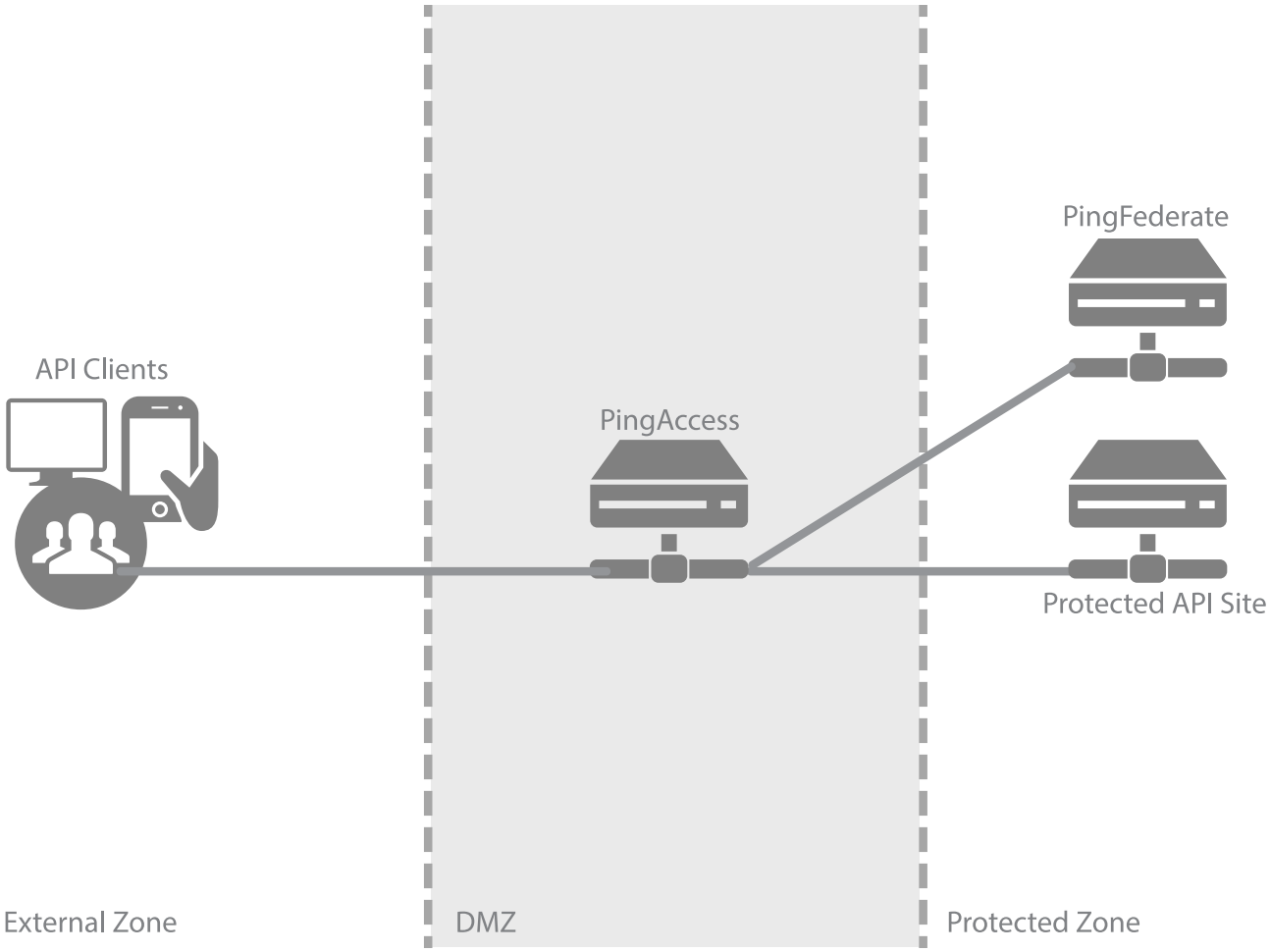
Zone	Description
External Zone	External network where incoming requests for web applications originate.
DMZ	Externally exposed segment where possibly multiple application web servers are accessible to web clients. PingAccess agent is deployed as a plugin on these web servers. Agents interact with PingAccess policy server in the protected zone.

Zone	Description
Protected Zone	Backend 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

The proof of concept environment emulates an API access management environment for testing purposes.

In the test environment, PingAccess can be set up with the minimum hardware requirements. Given these conditions, do not use this proposed architecture in a production deployment because 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 application programming interface (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	Backend 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.

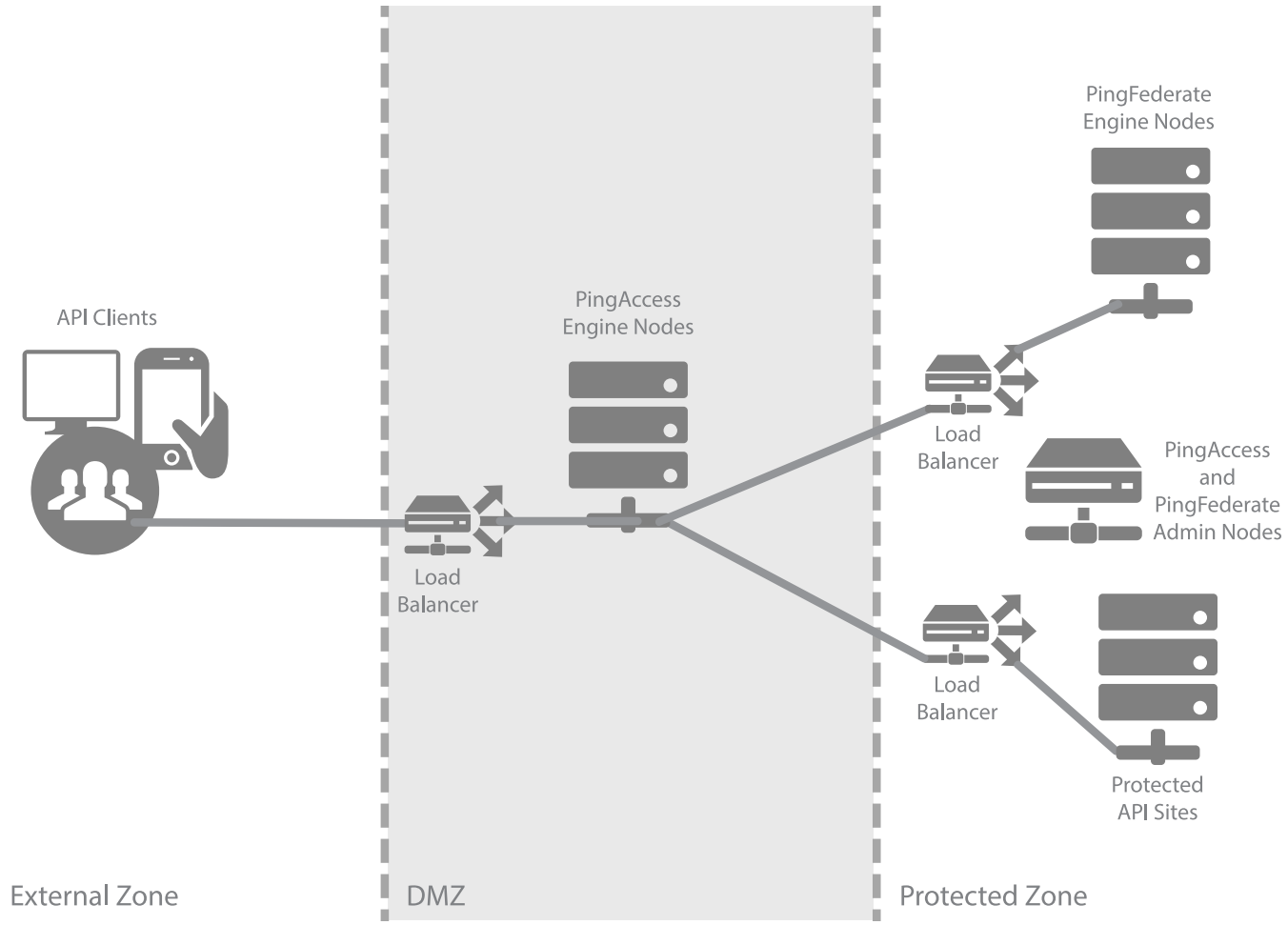
API access management production deployment architecture

This production deployment environment shows an API access 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.

> **Note**
>
> PingAccess provides high availability and basic load balancing for the protected web apps in the protected zone. For more information, see Managing load balancing strategies.

The following environment example is a recommended production quality deployment architecture for an API access management use case.



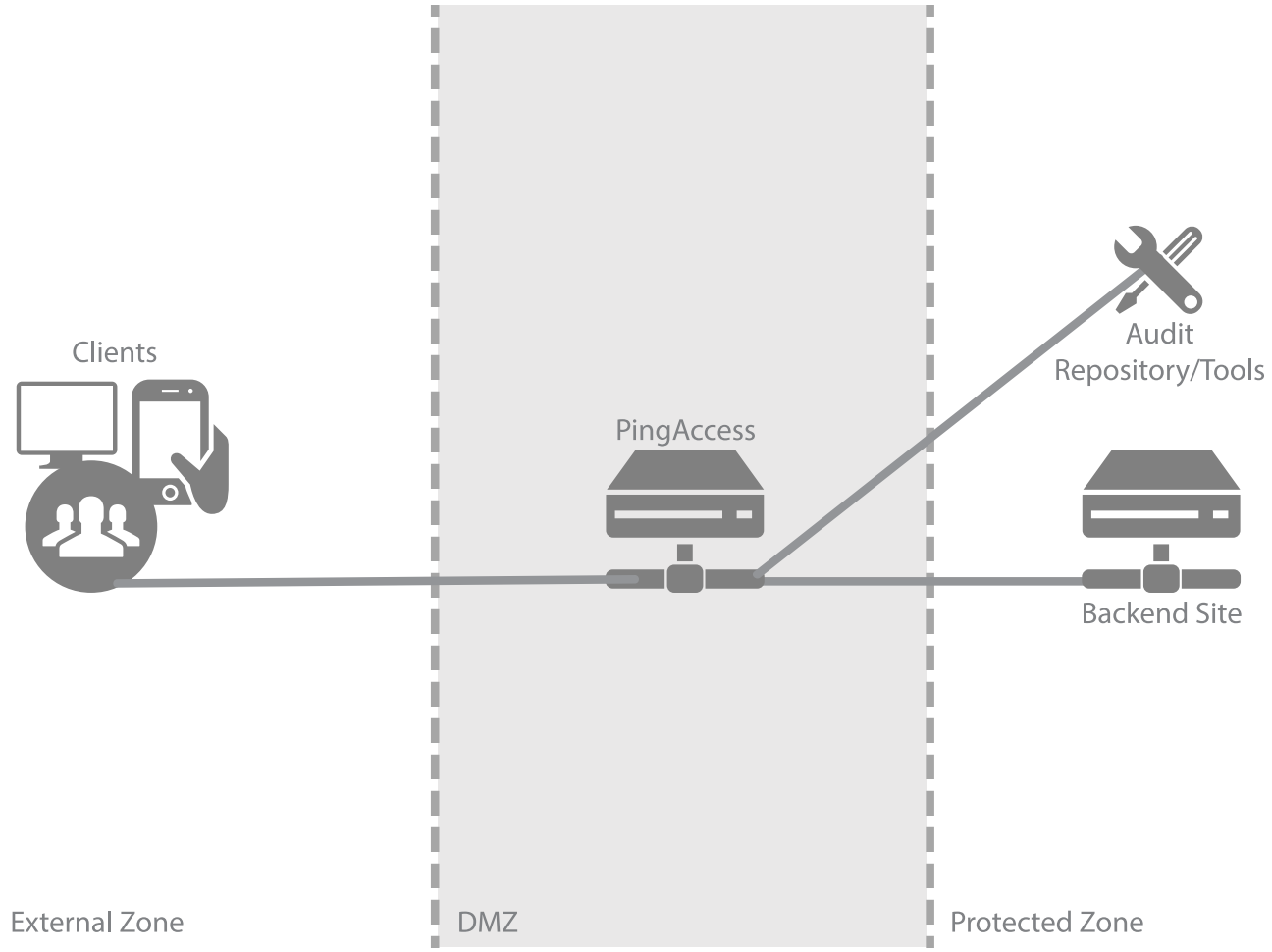
The following table describes the three zones within this proposed architecture.

External Zone	External network where incoming application programming interface (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, you might require more nodes.
Protected Zone	Backend 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 can be co-located on a single machine to reduce hardware requirements.

Auditing and proxying proof of concept deployment architecture

This proof of concept deployment environment is used to emulate an auditing and proxying environment for testing purposes in PingAccess.

In the test environment, you can set up PingAccess with the minimum hardware requirements. Given these conditions, do not use this proposed architecture in a production deployment because 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.

Zone	Description
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.

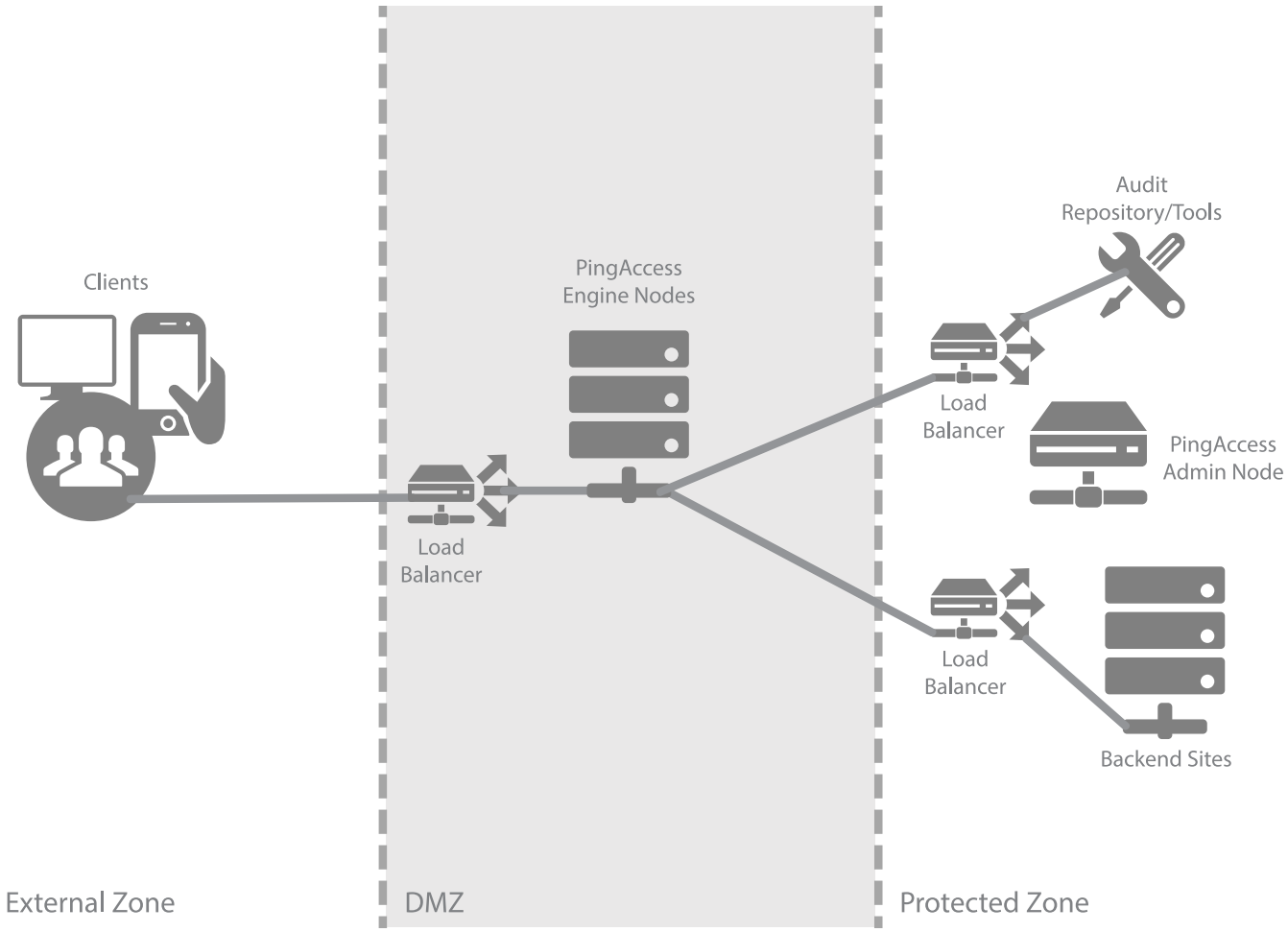
Auditing and proxying production deployment architecture

This production deployment environment shows an auditing and proxying architecture in PingAccess.

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.

> **Note**
>
> PingAccess can provide high availability and basic load balancing for the protected web apps in the protected zone. For more information, see Managing load balancing strategies.

The following environment example is a recommended production quality deployment architecture for an auditing and 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, you might require more nodes.
Protected Zone	Contains backend 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 in PingAccess

PingAccess provides the Groovy script and OAuth Groovy script rule types, which enable the use of Groovy, a dynamic programming language for the Java Virtual Machine (JVM).

Groovy scripts provide advanced rule logic that extends PingAccess rule development beyond the capabilities of the packaged rules. For more information, see the Groovy documentation.

Groovy scripts have access to important PingAccess runtime objects, such as the Exchange and PolicyContext objects, which the scripts can interrogate and modify.

Important

Groovy script rules and OAuth Groovy script rules must end execution with a matcher instance. For more information, see Matcher usage reference.
Groovy functions treat strings literally, and matchers perform case-sensitive string evaluation unless otherwise specified. For example, in the following line of code, the `caseSensitive` parameter determines whether the Groovy function performs case-sensitive comparison on the value.

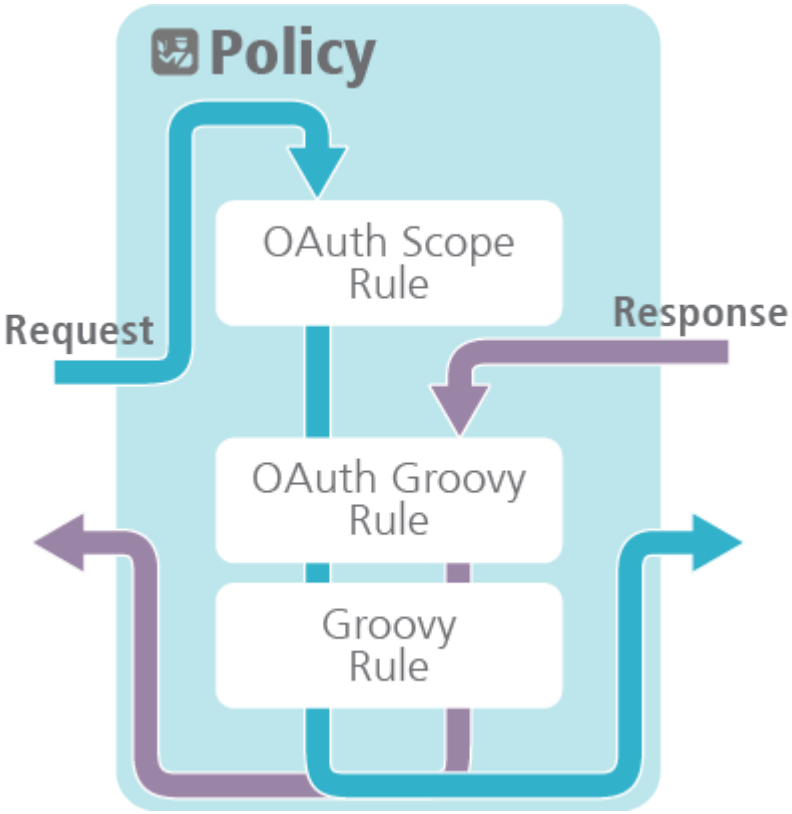
```
requestHeaderContains(Map<String, String> fieldValuesMap, boolean caseSensitive)
```

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.

Note

You can't access a mediated token through a Groovy rule because token mediation occurs after PingAccess rule processing.

The following diagram highlights the flow of rule processing.



1. During request processing, rules associated with the application are evaluated.
2. The request passes through each of the rules before PingAccess allows it to proceed.
3. 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 information 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. For more information on an object, click the link.

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.

> **Note**
>
> Some error messages about Groovy rule failures are only logged if `DEBUG` level output is enabled for the `com.pingidentity` logger.

Body object reference

This object accesses the Body object in Groovy `exc?.request?.body` or `exc?.response?.body`.

Purpose

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 reference

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. This object accesses the Exchange object in Groovy - `exc`.

Purpose

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. You can use the Exchange object 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 following Method Summary table 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.
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 Uniform Resource Identifier (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 reference

Access the Headers object in Groovy `exc?.request?.header` or `exc?.response?.header`.

Purpose

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 a site, as demonstrated in the following example:

ExampleGroovy 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. **Note** If you use Groovy Rules to inject HTTP headers for the backend protected application, the script must sanitize the same headers from the original client request.
`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.

Method	Description
`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<GroovyHeaderField> getAllHeaderFields()`	Returns a list of GroovyHeaderFields.
`String getHost()`	Returns the host name specified in the request.
`void setHost(String value)`	Sets the host name for the request to the Site.
`String getLocation()`	Gets the redirect location Uniform Resource Locator (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 Internet Protocol (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.
`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.

GroovyHeaderField object

Method summary

Method	Description
`String getValue();`	Returns the string's value.
`GroovyHeaderName getHeaderName();`	Returns the header's name.

ExampleGroovy sample

The following example demonstrates usage of the `getAllHeaderFields()` method, which includes both request and response logging:

```
exc?.log.info "Display Headers: "
exc?.log.info "-->Request Headers"
reqHdrs = exc?.request?.header?.getAllHeaderFields()
reqLoop = reqHdrs?.iterator()
while (reqLoop?.hasNext()) {
  hdr = reqLoop?.next()
  exc.log.info "-->reqHeader  Name: "+hdr?.getHeaderName()?.toString()
  exc.log.info "-->reqHeader Value: "+ hdr?.getValue()
}
exc?.log.info "-->Response Headers"
exc?.log.debug "-->Response HTTP Status: "+ exc?.response?.statusCode
rspHdrs = exc?.response?.header?.getAllHeaderFields()
rspLoop = rspHdrs?.iterator()
while (rspLoop?.hasNext()) {
  hdr = rspLoop?.next ()
  exc.log.info "-->rspHeader  Name: "+ hdr?.getHeaderName()?.toString()
  exc.log.info "-->rspHeader Value: "+ hdr?.getValue()
}
exc?.log.info "Display Headers EOF: "
pass()
```

Identity object reference

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 might 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 might change when new tokens are issued for the same identity.
`Date getTokenExpiration()`	Returns a Date object representing the time at which the authentication token expires. This might 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 reference

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.
`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.

Method	Description
`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 reference

This object accesses the Logger object.

Configuration

PingAccess must be configured to accept logging from Groovy rules.

In the `conf/log4j2.xml` file, uncomment or add the following line to enable debug-level logging from Groovy rules.

```
<AsyncLogger name="GroovyRule" level="DEBUG"/>
```

Uncomment or add the following line to enable info-level logging from the <*RuleName*> Groovy rule.

```
<AsyncLogger name="GroovyRule.<RuleName>" level="INFO"/>
```

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.

Method	Description
`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 reference

Access 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 reference

Access the Method object in Groovy `exc?.request?.method`.

Purpose

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 reference

Access the OAuth Token object in Groovy `policyCtx?.context.get("oauth_token")`.

Purpose

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, JSON Web Token (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 reference

Access the PolicyContext object in Groovy `policyCtx`.

Purpose

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 scripts rules.

Groovy sample

```
def oauthToken = policyCtx?.context.get("oauth_token")
```

Method summary

Method	Description
objectMap<String, Object> getContext()	Container for the OAuthToken.
Exchange getExchange()	Returns the exchange a message relates to.

Request object reference

Access the Request object in Groovy `exc?.request`.

Purpose

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?.header?.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 Uniform Resource Identifier (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** 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** 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 reference

Access the Response object in Groovy `exc?.response`.

Purpose

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 300s.
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 reference

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 server name indication (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

The following examples show possible uses for Groovy scripts.

OAuth Policy context example

In some instances, it might 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 through the `User` HTTP header. You can accomplish this 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`

This 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 whitelisting 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()
```

Combine an `AND` and `OR`, invoking an existing rule matcher

```
if ((anyOf(containsWebSessionAttribute("engineering", "true"),
containsWebSessionAttribute("marketing", "true")) && (containsWebSessionAttribute("manager",
"true")))
{pass()
}
else{
fail()
}
```

Matcher usage reference

Groovy script rules and OAuth Groovy script rules 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.

- For more information on Hamcrest, see the Hamcrest Tutorial.
- For more information on creating and troubleshooting Groovy scripts, and examples of how you might use Hamcrest matchers instead of PingAccess matchers, see Groovy Scripts.
- For more information on PingAccess matchers, review the following examples and tables.

> **Important**
>
> Matcher string evaluation is case sensitive unless otherwise specified. In the PingAccess matchers table, case insensitivity is called out in a matcher's description when applicable.

In the following example, the Groovy script rule inserts a custom HTTP header and the script ends with a call to the `pass()` matcher. The `pass()` matcher signals that the rule has passed.

```
test = "let's get Groovy!"
exc?.response?.header?.add("X-Groovy", "$test")
pass()
```

In the following example, the OAuth Groovy script rule checks the HTTP method and confirms the OAuth scope, and a matcher is evaluated at the end of each line of execution. The first matcher is the `hasScope()` matcher, which confirms whether the OAuth access token has the `WRITE` scope. If it does, the rule passes.

```
//Get the HTTP method name
def methodName = exc?.request?.method?.methodName()
if (methodName == "POST") {
    hasScope("WRITE")
} else {
    fail()
}
```

The `fail()` matcher combination is only evaluated when the `methodName` does not equal `POST`. This matcher combination evaluates to false.

PingAccess matchers

The following table lists the PingAccess 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.
`inIpRange(String cidr)`	Validates the source Internet Protocol (IP) address of the request against the `cidr` string 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 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 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")`

Matcher	Description
`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")`
`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)` **Note** This matcher can be case sensitive or case insensitive.	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)` **Note** This matcher can be case sensitive or case insensitive.	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 PingAccess token contains the attribute name and value. **Example:** `containsWebSessionAttribute("sub", "sarah")`
`containsACRValues(String value)`	Validates that the PingAccess token contains a matching ACR value.

The following table lists the PingAccess matchers available to only the OAuth Groovy script rule.

Matcher	Description
`hasScope(String scope)`	Validates that the OAuth access token contains the `scope` parameter. **Example:** `hasScope("access")`

Matcher	Description
`hasScopes(String… scopes)`	Validates that the OAuth access token contains the list of scopes. **Example:** `hasScopes("access","portfolio")`
`hasAttribute(String attributeName, String attributeValue)` **Note** This matcher is case insensitive and cannot be made case sensitive.	Checks for an attribute value within the current OAuth2 policy context. **Example:** `hasAttribute("account","joe")`

Performance tuning

While PingAccess has been engineered as a high performance engine, its default configuration might not match your deployment goals nor the hardware you have available. Use the recommendations here to optimize various aspects of a PingAccess deployment for maximum performance.

> **Note**
> The PingAccess capacity planning guide is also available to customers as a performance data reference. This document is available from the Customer Portal.

The features documented here are affected by the settings in the configuration file. For more information, see the Configuration file reference.

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, 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, increase the heap size.

For more information, see the following topics:

- Configuring JVM crash log in Java startup
- Configuring memory dumps in Java startup
- Modifying the Java heap size

Configuring JVM crash log in Java startup

Enable or disable the Java Virtual Machine (JVM) crash log.

About this task

The Java Virtual Machine (JVM) crash log is enabled by default. On Windows, the `run.bat` file specifies the JVM crash log location, and on Linux, the `run.sh` file specifies the JVM crash log location.

Steps

- Edit the `<PA_HOME>/bin/run.bat` file on Windows, or the `<PA_HOME>/bin/run.sh` file on Linux.

  Choose from:

  - 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"
    ```

  - 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"
    ```

Configuring memory dumps in Java startup

You can enable or disable Java Virtual Machine (JVM) memory dump, or change the memory dump's storage location.

About this task

The Java Virtual Machine (JVM) memory dump is disabled by default. On Windows, the `run.bat` file specifies the memory dump location, and on Linux, the `run.sh` file specifies the memory dump location.

Steps

- Edit `<PA_HOME>/bin/run.bat` on Windows, or `<PA_HOME>/bin/run.sh` on Linux.

  Choose from:

  - 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"
    ```

  - To disable JVM memory dump, comment out the line. For example:

    ```
    #HEAP_DUMP="-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=$PA_HOME/log"
    ```

Modifying the Java heap size

Modify the Java heap size for both Windows and Linux installations, including the Windows and Linux services.

Steps

1. Edit the `jvm-memory.options` file located in the `<PA_HOME>/conf` directory.
2. Specify overall heap size by modifying the `#Minimum heap size` and `#Maximum heap size` parameters.

   Choose from:

   - 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.

   Choose from:

   - 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`.

   > **Note**
   >
   > Not advisable if selecting the G1 collector. For more information, see Garbage Collector Configuration.
4. If you are running PingAccess as a Windows service, run the `generate-wrapper-jvm-options.bat` file located in the `<PA_HOME>/sbin/windows` directory.

   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 useful in preventing deployment issues in high capacity environments. For guidance specific to your operating system, see the following topics:

- Linux tuning
- Windows tuning

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.

For more information, see the following topics:

- Tuning network and TCP settings
- Increasing file descriptor limits (systemv)
- Increasing file descriptor limits (systemd)

Tuning network and TCP settings

Increase the performance and capacity of the networking, particularly TCP, stack to enable PingAccess to handle a high volume of concurrent requests.

Steps

1. Edit the `/etc/sysctl.conf` file.
2. Add or modify the following properties:

```
##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
###############
```

Increasing file descriptor limits (systemv)

Increase file descriptor limits in a systemv environment to enable PingAccess to handle a high volume of concurrent requests.

Steps

1. Edit the `/etc/security/limits.conf` file.
2. Add or modify the following lines:

```
<pingAccessAccount>  soft nofile  <value>
<pingAccessAccount>  hard nofile  <value>
```

`<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.

> **Note**
>
> 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.

Increasing file descriptor limits (systemd)

Increase file descriptor limits in a systemd environment to enable PingAccess to handle a high volume of concurrent requests.

Steps

1. Edit the `/etc/systemd/system/pingaccess.service` file.
2. Modify the following line under the `[Service]` section:

```
LimitNOFILE=<value>
```

<*value*> is the new value. The default value of 65536 (64K) should be sufficient for most environments.

> **Note**
>
> 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 can 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.

3. Run the following command as root:

```
systemctl daemon-reload
```

4. Restart the PingAccess service.

Windows tuning

This section describes tuning recommendations for the Windows operating system environment, version 7 or later.

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.

For more information, see the following topics:

- Increasing the number of available ephemeral ports
- Reducing the socket TIME_WAIT delay

Increasing the number of available ephemeral ports

Increase the number of available ephemeral ports to prevent deployment issues, particularly in high capacity environments.

About this task

This setting increases the performance and capacity of network, specifically the TCP socket, connectivity, enabling PingAccess to handle a high volume of concurrent requests.

Steps

1. View the ephemeral ports with the `netsh int ipv4 show dynamicportrange tcp` command.
2. Increase the ephemeral ports with the `netsh int ipv4 set dynamicport tcp start=1025 num=64510` command.
3. Reboot the machine.
4. View and confirm the updated port range with the `netsh int ipv4 show dynamicportrange tcp` command.

Reducing the socket TIME_WAIT delay

Reduce the socket `TIME_WAIT` delay to prevent deployment issues, particularly in high capacity environments.

About this task

This setting increases the performance and capacity of network, specifically the TCP socket, connectivity, enabling PingAccess to handle a high volume of concurrent requests.

Steps

1. Click **Start → Run**.

2. Type `regedit` and click **OK** to open the registry editor.
3. Go to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters`.
4. Create a new `DWORD` value, 32 bit, and provide the name `TcpTimedWaitDelay`.
5. Set a decimal value of `30`.
6. Reboot the machine.

Garbage collector configuration reference

The following 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 the `<PA_HOME>/conf` directory. 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 Java Virtual Machine (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 the `jvm-memory.options` file.
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 the `jvm-memory.options` file. 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.

Configuring acceptor threads

Configure the pool of acceptor threads based on your environment.

About this task

PingAccess uses a pool of threads to respond to HTTP/S requests made to the TCP ports 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 can be used.

To modify the pool of acceptor threads:

Steps

1. Open the `run.properties` file located in the `conf` directory of your PingAccess deployment.
2. Specify the number of acceptors you want to use on the following lines, where <*N*> represents the number of acceptor threads:
    - `admin.acceptors=<N>`
    - `engine.http.acceptors=<N>`
    - `agent.http.acceptors=<N>`

Configuring worker threads

Modify the minimum and maximum number of worker threads to increase performance.

About this task

PingAccess engines use a pool of worker threads to process requests received on the runtime ports, and another pool to process requests from agents. These worker threads do the initial handling of request processing within PingAccess. By default, PingAccess starts with a minimum of 5 worker threads in each pool. New threads are created as needed, with the maximum size being unlimited.

Maintenance of the pool size is such that if the number of threads in the pool exceeds the value of `engine.httptransport.coreThreadPoolSize` or `agent.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. The same is true for `agent.httptransport.coreThreadPoolSize` and `agent.httptransport.maxThreadPoolSize`.

Unless you want to restrict the maximum number of concurrent requests admitted for processing by a PingAccess engine, you should leave `engine.httptransport.maxThreadPoolSize` and `agent.httptransport.maxThreadPoolSize` set to unlimited (-1).

For `engine.httptransport.coreThreadPoolSize` and `agent.httptransport.coreThreadPoolSize`, the defaults should be sufficient for all deployments. The creation of new threads is efficient, so it is unlikely that end users could detect any improvement in performance by increasing the values for these properties. Increasing these values also increases the baseline memory footprint of PingAccess due to the increase in memory for the additional threads.

These worker threads do not perform any input or output processing. Incoming and outgoing communication (I/O) is performed by two additional pools of threads. One pool handles request from, and responses to, clients; another handles requests to, and responses from, application servers proxied by PingAccess. These pools use a fixed number of threads and by default, the number of threads in these pools is equal to twice the number of logical CPU cores on the host computer.

To modify the number of worker threads:

Steps

1. Open the `run.properties` file located in the `conf` directory of your PingAccess deployment.
2. Add or edit the following properties, where <*N*> represents the number of worker threads:

   ```
   engine.httptransport.maxThreadPoolSize=<N>
   ```

   and

   ```
   agent.httptransport.maxThreadPoolSize=<N>
   ```

Backend server connections

PingAccess provides a max connections option to control and optimize connections to the proxied site.

Maximum 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 website. If, for example, the website 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 the 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.

> **Note**
>
> It is important to understand the limits and tuning of the server application being proxied. Setting the **Maximum Connections** value too low might create a bottleneck to the proxied site, setting the value too high, or unlimited, might cause PingAccess to overload the server.

For information on setting the **Maximum Connections**, see Sites documentation.

Logging and Auditing

PingAccess uses a high performance, asynchronous logging framework to provide logging and auditing services with the lowest possible impact to overall application performance.

Logging

Modify your logging settings to increase performance.

Although logging is handled by a high performance, asynchronous logging framework, it is more efficient for the system overall to log the minimum amount of information required. Review the logging section of the documentation and adjust the logging level to the lowest level that suits your needs.

Auditing

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. 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), audit to a database. You will see a decrease in disk I/O, which might result in increased performance, depending on database resources.

Agent tuning reference

Modify the properties of your PingAccess agents to improve performance.

You can configure several properties in the `agent.properties` file for increased performance. For more information on agent configuration and setting properties, see the agent documentation for Apache or IIS.

Maximum Connections

Connections from the agent to PingAccess are limited by the `agent.engine.configuration.maxConnections` property. Though the default value 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 the 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**
>
> 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 might create a bottleneck to PingAccess, and setting the value too high might cause PingAccess to become overloaded.

Maximum 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

Use this guide to configure PingAccess features and components. For more comprehensive instructions on configuring your PingAccess implementation, use this guide alongside PingAccess's use case documentation.

For ease of use, this guide is modeled after the PingAccess administrative console layout:

- For more information about PingAccess's features and functions, see the Introduction to PingAccess.
- The features documented in the PingAccess user interface reference guide are affected by settings in the configuration file. For more information, see the configuration file reference guide.

> **Note**
>
> To learn more about configuration options for a particular window, see its corresponding topic.

Use filters in the PingAccess administrative console to control the entries displayed in your list of configured PingAccess components. The following tables list all of the filters in the PingAccess administrative console.

Filter	Page	Description
All Sites, Agents and Sideband Clients	**Applications**	Filters your configured applications by the sites, agents, or sideband clients that you've assigned them to, if applicable.
All Virtual Hosts	**Applications**	Filters your configured applications by the virtual hosts that you've assigned them to. The default virtual host options are `3000*` and `localhost:3000`.
All SPA Support Status	**Applications**	Filters your configured applications by their SPA support status: ***All SPA Support Status*** Display all applications regardless of whether they have SPA support enabled. ***Enabled*** Display only applications that have SPA support enabled. ***Disabled*** Display only applications that don't have SPA support enabled.

Filter	Page	Description
Any Rule Type	**Rules**	Filters your configured rules by rule type. For more information on the configurable rule types, see Rule management.
Any Application	**Rules** **Rule Sets** **Rule Set Groups**	Filters your configured Rules, Rule sets, or Rule set groups by the application that you assigned them to, if applicable.

> **Note**
>
> Filters won't appear on the **Rules**, **Rule Sets**, or **Rule Set Groups** pages until you configure a rule, rule set, or rule set group, respectively.

Filter	Page	Description
Any HTTPS Listener	**Key Pairs**	Filters your configured Key pairs by the HTTPS listener that you've assigned them to. For more information on the types of HTTPS listeners, see Assigning key pairs.
Any Virtual Host	**Key Pairs**	Filters your configured key pairs by the virtual host that you've assigned them to. The `3000*` default virtual host isn't included in this filter.

Filter	Page	Description
Any ACME Status	**Key Pairs**	Filters your configured key pairs by their current ACME status: ***Any ACME Status*** Display all key pairs regardless of whether you've configured ACME management for them. ***ACME Invalid*** Displays key pairs that you've configured ACME management for only if the ACME protocol didn't complete successfully. ***ACME Challenge Pending*** Displays key pairs that you've configured ACME management for only if the ACME protocol hasn't completed yet. ***Managed by ACME*** Displays key pairs that you've configured ACME management for only if the ACME protocol completed successfully.
Any HSM	**Key Pairs**	Filters your configured key pairs by the hardware security module (HSM) that you've created them on: ***Any HSM*** Displays all key pairs regardless of whether you've created them on an HSM. ***HSM*** Displays only key pairs that you've created on an HSM. ***No HSM*** Displays only key pairs that you haven't created on an HSM.

Applications header

The Applications header contains menu options related to directly administering sites and applications.

The applications header contains these menu options:

- Applications
- Sites
- Agents
- Sideband Clients

Applications

This section contains controls for managing applications, resources, and redirects.

Choose from one of the following sub-sections:

- Applications operations
- Global unprotected resources
- Redirects
- Virtual hosts

Applications operations

Applications represent the protected web applications and APIs that receive client requests.

Applications consist of one or more resources, have a common virtual host and context root, and correspond to a single target site. Applications use a common web session and identity mapping. Apply access control and request processing rules and their resources on the **Policy Manager** window to protect them. Applications can be protected by a PingAccess gateway or a 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 three application types:

- Web
- API
- Web + API

Web + API applications allow administrators to configure both Web and application programming interface (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 fall back to the method you specify as the fallback type for the application.

Use the **Policy Manager** window to define the applications which PingAccess protects and to which client requests are ultimately forwarded. 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 applies to API and Web + API applications. 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 are properly redirected to authenticate by the same response. For Web applications, authentication challenge responses fulfill the same role. See Authentication for more information.

It might 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 can 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 for authentication, rules, or identity mapping.

For additional guidance in preparing a SPA to work with PingAccess, see the SPA developer's guide in the PingAccess resources on github. This guide contains a sample application before and after onboarding.

Adding an application

Add a new application in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click **+ Add Application**.
3. Complete the fields.

   Learn more in Application field descriptions.
4. Click **Save**.

   Save & Go to Resources lets you configure additional application resources. Learn more in Adding application resources.

> **Note**
>
> When you save the application, PingAccess verifies that the redirect Uniform Resource Identifier (URI) for the application's virtual host is configured in PingFederate. If PingAccess determines that the redirect URI is not defined, you will see 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 your configuration. If you have a wildcard in your virtual host configuration, ensure the redirect URI list includes the same wildcard host definition, otherwise you might 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

The following table describes the fields available for managing applications on the **Applications** tab.

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 application programming interface (API).
Case Sensitive Path	No	Indicates whether or not to make request Uniform Resource Locator (URL) path matching case sensitive.

Field	Required	Description
Virtual host(s)	Yes	Specifies the virtual host for the application. Click **+ Create** to create a virtual host. See Creating new virtual hosts for more information.

Application Type	Yes	Specifies the application type, either `Web`, `API`, or `Web + API`. • If the **Application Type** is `Web`, select the **Web Session** if the application is protected and, if applicable, the **Web Identity Mapping** for the application. Select an **Authentication Challenge Policy** to produce authentication challenges for the application. You can enter an **OpenID Connect Provider Issuer URL** to replace the visible URL during authentication if the token provider supports it. Select a **Risk Policy** to enforce continuous authorization with PingOne Protect on the application. **Note** You must set up a PingOne connection before you can create a risk policy. For more information on how to set these up through the administrative console, see PingOne Protect integration, Adding a PingOne connection and Adding a risk policy. PingOne risk policies depend on mapping user identity attributes to the risk evaluation requests, so the PingAccess administrative console will prevent you from saving an unprotected application or resource with a risk policy. Click **+Create** underneath the desired field to create a new web session, identity mapping, authentication challenge policy, or risk policy. • 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 **API Identity Mapping** for the application. If you want to configure different DPoP settings for the application than the global settings you specified in the token provider, follow step 9 in Adding application resources to complete your configuration. Click **+ Create** to create an access validation or identity mapping.

Field	Required	Description
		Note If you try to use a remote access token validator on your PingAccess API application without first configuring the introspection endpoint on the **OAuth Authorization Server** tab of the **Token Provider** page, you get the following error message: `Cannot use remote validation as authorization server does not have an Introspection endpoint.` • If the **Application Type** is `Web + API`, indicate the method of **Access Validation**. Select the **Web Session** and, if applicable, the **Web Identity Mapping** and **API Identity Mapping** to use for each type. Select an **Authentication Challenge Policy** to produce authentication challenges for the application. In this configuration, the web session is required and the API is protected by default. Click **+ Create** to create an access validation, web session, web identity mapping, API identity mapping, or authentication challenge policy. You can enter an **OpenID Connect Provider Issuer URL** to replace the visible URL during authentication if the token provider supports it. Specify whether you want to enable SPA Support. If you want to configure different DPoP settings for the application than the global settings you specified in the token provider, follow step 9 in Adding application resources to complete your configuration.

Field	Required	Description
Destination	Yes	Specifies the application destination type, either `Site`, `Agent`, or `Sideband`. • 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. For more information, see Adding sites. • If the destination is an `Agent`, select the agent that intercepts and validates access requests for the application. Click **+ Create** to create an Agent. For more information, see Adding agents. • If the destination is `Sideband`, select the sideband client that queries PingAccess for authorization and request/response modification. Click **+ Create** to create a sideband client. For more information, see Adding sideband clients.
Enabled	No	Select to enable the application and allow it to process requests.

Advanced Settings

To configure advanced settings on an application, expand the **Show Advanced Settings** section at the bottom of the **Applications** tab, just above the **Enabled** check box. These settings are optional.

Field	Description
Use context root as reserved resource base path	Selecting the **Use context root as reserved resource base path** check box provides access to reserved PingAccess resources and runtime API endpoints from the context root of this application in addition to the globally-defined reserved application context root. **Note** By default, the reserved application context root is `/pa`. You can change this value in `/applications/reserved` using the PingAccess admin API. If you change the reserved application context root, make sure to update your calls to PingAccess endpoints and any other application URLs that reference the reserved path accordingly. When this setting is enabled, PingAccess adds this application's context root before the reserved application context root in any content that references the reserved path to ensure that it responds to those requests. For example, the path to an endpoint changes from `[reserved application context root]/[endpoint]` to `[application context root]/[reserved application context root]/[endpoint]`. If you have a web application with a context root of `myApp`, PingAccess changes the path to the OpenID Connect (OIDC) endpoint to `https://[host]/myApp/pa/oidc/cb` instead of `https://[host]/pa/oidc.cb`.

Editing an application

Edit an existing application in PingAccess.

About this task

Tip

To find a specific type of application efficiently, expand **Filters** next to the search bar on the **Applications** page. You can filter your search results by destination type, virtual host, or SPA support status.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click the **Expand** icon to see more details about the application that you want to edit.
3. On the **Properties** tab, click the **Pencil** icon.
4. Make the required changes.

For more information, see Application field descriptions.

5. To confirm your changes, click **Save**.

Deleting an application

Delete an existing application in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click to expand the application you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Authentication challenge responses

This table describes the authentication challenge responses generated by PingAccess, based on its configuration and properties of the request.

An authentication challenge response is an HTTP response sent to a user agent (such as a web browser) by PingAccess, telling the user agent that the corresponding request did not contain a valid authentication token. Some responses also provide instructions to the user agent to obtain a valid authentication token such as an HTTP redirect response containing an encoded OpenID Connect (OIDC) authentication request.

When onboarding new applications to PingAccess, the recommended configuration is `SPA Support = Enabled`, `Request Preservation = POST and Fragment`, and `Fail on Unsupported Content Type = false`, regardless of the behavior of the application. This configuration is displayed in the first table.

Recommended configurations

PingAccess configuration			Request properties			Response characteristics	
SPA Support[1]	**Request Preservation**[2]	**Fail on Unsupported Content Type**[3]	**Method**	**Content Type**	**Accept Header Field**	**Response Code**	**Body Content**
Enabled	POST, POST and Fragment	Any	GET[4]	Any	NOT application/ json	401	HTML
Enabled	POST, POST and Fragment	Any	GET[4]	Any	application/ json	401	JSON

PingAccess configuration			Request properties			Response characteristics	
Enabled	POST, POST and Fragment	false	POST	Any	NOT application/json	401	HTML
Enabled	POST, POST and Fragment	false	POST	Any	application/json	401	JSON

[1]Configured on an application. In the Admin application programming interface (API), the field is **spaSupportEnabled**. In the UI, the field is **SPA Support**. See Adding an application for more information about this field.

[2]Configured on a web session. In the Admin API, the field is **requestPreservationType**. In the UI, the field is **Request Preservation**. See Creating web sessions for more information about this field.

[3]This option is only available through the Admin API.

[4]Any non-POST method receives the same response as a GET.

Additional configurations

PingAccess configuration			Request properties			Response characteristics	
SPA Support[1]	**Request Preservation**[2]	**Fail on Unsupported Content Type**[3]	**Method**	**Content Type**	**Accept Header Field**	**Response Code**	**Body Content**
Disabled	None	Any	Any	Any	Any	302	None
Disabled	POST	Any	GET[4]	Any	Any	302	None
Disabled	POST	Any	POST	application/x-www-form-urlencoded	Any	200	HTML
Disabled	POST	false	POST	NOT application/x-www-form-urlencoded	Any	302	None
Disabled	POST	true	POST	NOT application/x-www-form-urlencoded	Any	415	HTML

PingAccess configuration			Request properties			Response characteristics	
Disabled	POST and Fragment	Any	GET[4]	Any	Any	200	HTML
Disabled	POST and Fragment	Any	POST	application/x-www-form-urlencoded	Any	200	HTML
Disabled	POST and Fragment	false	POST	NOT application/x-www-form-urlencoded	Any	302	None
Disabled	POST and Fragment	true	POST	NOT application/x-www-form-urlencoded	Any	415	HTML
Enabled	None	Any	Any	Any	NOT application/json	401	HTML
Enabled	None	Any	Any	Any	application/json	401	JSON
Enabled	POST, POST and Fragment	true	POST	NOT application/x-www-form-urlencoded	NOT application/json	415	HTML
Enabled	POST, POST and Fragment	true	POST	application/x-www-form-urlencoded	NOT application/json	401	HTML
Enabled	POST, POST and Fragment	true	POST	Any	application/json	401	JSON

Application resources

Application resources are components in an application that require a different level of security. You can manage security settings for application resources.

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, you can configure manual resource ordering to 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 can have different policy requirements. Manual 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.

Configuring resource ordering in PingAccess

Enable and disable resource ordering in PingAccess to control how requests are processed.

About this task

Application resources are defined by one or more path patterns and zero or more query parameters. When handling requests, PingAccess matches the path pattern and query parameters with the proper resource. When more than one resource matches a request, PingAccess uses the first matching resource it identifies.

Manual resource ordering allows you to specify which resources are checked for a match in what order, letting you control the policy that is applied to a particular request. These instructions describe how to configure and disable resource ordering in PingAccess, as well as how to use the auto-ordering feature.

> **Note**
>
> Resources can only include query parameters if manual resource ordering is configured.

Steps

1. To configure manual resource ordering:
    1. Edit your application.
    2. On the **Resources** tab, click **Order Resources**.
    3. Modify the resource order by dragging resources or by entering a new position for each resource.

4. **Optional:** Click **Auto Order** to reset the resource order to the order that would be used under automatic ordering.
5. Click **Save**.

2. To disable manual resource ordering:
    1. Edit your application.
    2. On the **Resources** tab, remove any Regex path patterns.
    3. Clear the **Manual ordering mode** check box.
    4. Click **Save**.

Adding application resources

Add application resources to existing applications in PingAccess.

About this task

An application resource is a component within an application that requires a different level of security. These instructions describe how to add, edit, and delete application resources and how to configure resource ordering, authentication policy, and application type.

There are two resource types: standard and virtual.

Standard resources

Standard resources exist on the target destination, and users can be directed to them.

Virtual resources

Virtual resources exist only in PingAccess. When a user attempts to access them, PingAccess generates a specified response.

> **Note**
>
> Some applications allow you to specify the parameters of a request in the query string or the POST body. If you are managing such an application and are defining its resources using query parameters, use caution when defining the resource to ensure that PingAccess and the application treat the resource in the same way.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click to expand the application that you want to modify resources on, then click the **Pencil** icon and click the **Resources** tab.

    Choose from:

    - To add a resource, click **Add Resource**.

> **Note**
>
> A group containing all global unprotected resources is displayed on the first **Resources** page. Review this list before adding a resource to make sure that there won't be a conflict between the new resource's path patterns and any unprotected resource path pattern.

- To edit a resource, expand the resource and click the **Pencil** icon.
- To delete the resource, expand the resource and click the **Delete** icon.

3. Enter a unique **Name** of up to 64 characters, including special characters and spaces.
4. Enter a list of Uniform Resource Locator (URL) path patterns within the context root that identify this resource.

   If resource ordering is enabled, select the path pattern type, **Basic** or **Regex**.

   The path pattern must start with a forward slash (/). It begins after the application context root and extends to the end of the URL.

   If automatic path pattern evaluation ordering is in use (default), patterns can contain one or more wildcard characters (`*`). No use of wildcards is assumed. For example, there is a difference between `/app/` and `/app/*`. If a request matches more than one resource, the most specific match is used.

   If you enable manual path pattern ordering (resource ordering), you can use regular expressions in the path pattern. When one or more path patterns contain a regular expression, you cannot revert to automatic path pattern ordering unless you remove that path pattern

   > **Note**
   >
   > If you have specified a regular expression, make sure that you select the **Regex** path pattern type. If you don't, the pattern will be interpreted incorrectly as a **Basic** text string.
   > 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` application programming interface (API).
   > If you selected the **Use context root as reserved resource base path** check box on your application, you cannot use the application context root as a path pattern either. This check box turns the application context root into a reserved resource, and a reserved application context root cannot be used in an application's resource paths.

5. If you have enabled resource ordering, select an option in the **Query Parameters** section.

   The **Query Parameters** section lets you define the resource by query parameters in addition to path patterns.

   Choose from:

   - To define the resource without regard to query parameters, select **Match Any**.
   - To define the resource using one or more query parameters, select **Match Specific**:
     - Select **Matches No Parameters** to match the result to the resource if no query parameters are present and if at least one query parameter is present and matches. If this option is deselected, at least one query parameter must be present and must match.
     - Enter one or more **Name**-**Value** pairs, or enter a **Name** and select **Any** to match any value for the given name.

6. Select a **Resource Authentication** type.

   Choose from:

   - If the resource requires the same authentication as the root application, select **Standard**.
   - If this resource has no authentication requirements, select **Anonymous**. Identity mappings are still applied if the user is already authenticated. Access control and processing rules are applied where applicable.
   - If this resource has no authentication requirements, select **Unprotected**. 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 are unprotected when they don't have an associated web session.
   > - API applications are unprotected when they aren't protected by an authorization server.

7. If the application is a protected web application with a web session, select an **Authentication Challenge Policy** to generate authentication challenge responses for the resource or click **+ Create** to create a new authentication challenge policy. Otherwise, proceed to the next step.

8. If you are using the PingOne Protect integration on a protected web application, select a **Risk Policy** to enforce on the resource or click **+Create** to create a new risk policy. Otherwise, proceed to the next step.

   > **Important**
   >
   > You must set up a PingOne connection before you can create a risk policy. For more information on how to set these up through the administrative console, see Adding a PingOne connection and Adding a risk policy. For more information on the capabilities that a risk policy can provide, see PingOne Protect integration.
   > A risk policy applied at the resource level takes precedence over one applied at the application level. PingOne risk policies depend on mapping user identity attributes to the risk evaluation requests, so the PingAccess administrative console will prevent you from saving an unprotected application or resource with a risk policy.

9. **Optional:** If you're using an **API** or **Web + API** type application and want to configure different DPoP settings for the application resource than the global DPoP settings that you configured in the token provider:

   > **Note**
   >
   > If you are using PingFederate as the token provider, you must use PingFederate 11.3 or later to configure DPoP support.

   1. Select **Override DPoP Settings**.
   2. In the **DPoP Type** list, select the level of OAuth 2.0 Demonstrating Proof of Possession (DPoP) support that you want to enable for access token validation:
      - **Off** (default): PingAccess doesn't accept DPoP-bound access tokens, only bearer tokens.
      - **Enabled**: PingAccess accepts both bearer tokens and DPoP-bound access tokens.
      - **Required**: PingAccess doesn't accept bearer tokens, only DPoP-bound access tokens.

3. To require each DPoP proof to contain a nonce value during validation that was provided by PingAccess when the access token was created, per RFC 9449 section 9, select **Require Nonce**.

   This check box is cleared by default.

4. In the **DPoP Proof Lifetime (SEC.)** field, enter the duration, in seconds, that a DPoP proof should be considered valid after it's issued.

   > **Important**
   >
   > As a security best practice, keep this value low and consistent with the DPoP implementation of your API client. The default value is 120 seconds.

10. In the **Methods** list, select one or more methods supported by the resource.

    For a complete list of HTTP methods and their definitions, see https://webconcepts.info/concepts/http-method/.

    > **Note**
    >
    > PingAccess does not support the CONNECT, PRI, or QUERY methods.

    Leave the asterisk default if the resource supports all other HTTP methods, including custom methods.

    Defining methods for a resource allows more fine-grained access control policies on resources. If you have a server optimized for writing data (POST, PUT) and a server optimized for reading data (GET), you might want to segment traffic based on the operation being performed.

11. To log information about the transaction to the audit store, select the **Audit** check box.

12. 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**
    >
    > 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 relies on this configuration to determine which processing flow to use.

13. To enable the resource, select the **Enabled** check box.

14. In the **Resource Type** list, select a resource type:

    Choose from:

    - If the resource exists on the target destination, select **Standard**.

- If the resource only exists in PingAccess, select **Virtual**. PingAccess generates a response when a user attempts to access the resource.

15. If you selected the **Virtual** resource type, in the **Type** list, select a response generator type:

    Choose from:

    - To redirect the user to a new URL with the specified response code, select **Redirect**.
    - To create a response using a specified template, select **Template**.
    - To make user attributes available to other applications as a JavaScript Object Notation (JSON) payload, select **JSON Identity Mapping**.
    - To end the application web session and optionally redirect the user to a specific landing page after logout, select **Logout**. All applications using the same web session are logged out.

      > **Note**
      >
      > You can only use the **Logout** virtual resource type if PingFederate is the configured token provider.

16. If you selected the **Redirect** response generator, specify the redirect parameters:

    1. In the **Redirect URL** field, enter a relative or absolute URL to which users should be redirected.
    2. In the **Response Code** list, select a response code:

       301 – Moved permanently

       This is a permanent redirect that does not require the redirect to maintain the original HTTP method.

       302 – Found

       This is a temporary redirect that does not require the redirect to maintain the original HTTP method.

       307 – Temporary Redirect

       This is a temporary redirect that requires the redirect to maintain the original HTTP method.

       308 – Permanent Redirect

       This is a permanent redirect that requires the redirect to maintain the original HTTP method.

    3. To opt out of automatic URL encoding, deselect the **Encode URL** check box.

       Learn more in PingAccess 8.1 (June 2024).

17. If you selected the **Template** response generator, specify the template parameters:

    1. In the **Media Type** list, select or enter a media type for the template.
    2. In the **Template** field, enter a template in Velocity Template Language (VTL).

       When a user accesses the virtual resource, the template is processed and returned as the response.

       The template can include information about the user, resource, and application according to this data model:

`identity.subject`

A string containing the subject name of the identity. This property is only available if the user is authenticated.

`identity.attributes`

An object containing user attributes set by the token provider. For example, `identity.attributes.role` could contain a role set by the token provider. This property is only available if the user is authenticated.

`identity.trackingID`

A string containing the tracking ID of the identity. This property is only available if the user is authenticated.

`resource.name`

A string containing the name of the requested resource.

`application.name`

A string containing the name of the requested application.

`application.realm`

A string containing the OAuth realm associated with the application. If the realm is not defined by the application, it is inferred to be the requested authority and the application's context root.

`exchangeId`

A string containing the ID for the current transaction.

3. In the **Response Code** list, select a response code:
    - 200 – OK
    - 201 – Created
    - 400 – Bad Request
    - 401 – Unauthorized
    - 403 – Forbidden
    - 404 – Not Found
    - 405 – Method Not Allowed

18. If you selected the **JSON Identity Mapping** response generator, select **Inclusion List** or **Exclusion List**:

    Choose from:

    - To map the specified attributes to corresponding property names, select **Inclusion List**. If you select this option, enter a corresponding **Attribute Name** and **Property Name** on each row. Click **+ Add Row** to add additional rows.
    - To expose all attributes except for those you specify, select **Exclusion List**. If you select this option, enter zero or more excluded attributes in the **Excluded Attributes** field.

19. If you selected the **Logout** response generator, specify the logout parameters:
    1. **Optional:** In the **Post-logout Redirect URI** field, enter a Uniform Resource Identifier (URI) to which the user is directed after logout. The format of this URI determines the logout behavior:
        - No URI: single logout (SLO) defaults to the token provider settings.
        - Absolute URL without variables: The PingAccess session is cleared and SLO is not triggered.
        - URL containing the `${SLO}` variable: The `${SLO}` variable is replaced with the PingFederate `ping_end_session_endpoint`, which triggers SLO.

          For example, if the PingFederate `ping_end_session_endpoint` is `https://pingfederate:9031/idp/startSLO.ping`, a value of `${SLO}?TargetResource=https://example.com` would direct the user to the PingFederate endpoint, trigger SLO, and then redirect the user to `https://example.com`.
        - Relative path: The relative path is appended to the application path to form the destination and SLO isn't triggered.
        - PingFederate parameters: The parameters are passed to PingFederate and SLO is triggered. For more information, see IdP endpoints[IdP endpoints].
    2. To opt out of automatic URL encoding, deselect the **Encode URL** check box.

       Learn more in PingAccess 8.1 (June 2024).
20. Click **Save**.

Path patterns reference

PingAccess uses application resource path patterns to match resources. This reference describes the two path pattern types used by PingAccess and how they are processed.

For more flexible resource matching, PingAccess supports two types of path matching patterns:

- Basic
- Regex

To specify a path pattern as Basic or Regex, 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.

Example

```
/path/x/*
```

matches any of the following request paths:

```
/path/x/
/path/x/index.html
/path/x/y/z/index.html
```

Regex patterns

Regex path pattern support occurs when you enable resource ordering.

> **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.
> Regex path patterns allow for more flexibility in resource matching.

Example

```
/[^/]+/[a-z]+\.html
```

matches any of these request paths:

```
/images/gallery.html
/search/index.html
```

However, it would not match any of these paths:

```
/images/gallery2.html
/search/pages/index.html
/index.html
```

The supported syntax for Regex patterns is documented by the RE2 wiki.

Use of Regex path patterns in agent deployments:

- Though Regex path patterns function in an agent deployment, a performance decrease might occur 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.

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, therefore a performance decrease might occur.

> **Tip**
>
> If you are using Regex path patterns in an agent deployment, and the order in which resources are ordered is unimportant, order Regex path patterns at the end of the list. If the order is important, place the resource where appropriate to ensure the correct policy is 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, consider redeploying these applications in a proxy configuration where possible.

Applying rules to applications and resources

Apply rules, rule sets, and rule set groups to applications and resources in PingAccess.

About this task

You can apply application access control and request processing rules to applications and their resources. These instructions describe how to create, apply, organize, and remove application rules.

Steps

1. Click **Applications**, then go to **Applications > Applications**.
2. Click to expand an application in the list.
3. Click the **Pencil** icon.
4. **Optional:** Manage the policies for a resource.
    1. Click the **Resources** tab.
    2. Click to expand the resource you want to edit.
    3. Click the **Pencil** icon.
    4. Make the desired changes to the resource.
    5. To confirm your changes, click **Save**.

5. Select the applicable tab.

   Choose from:

   - 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.

6. Using the radio selection, filter by Rules, Rule Sets, Rule Set Groups, or **Rule Type**.
7. To create a new rule, click **Create Rule**.
8. To apply a rule, rule set, or rule set group, drag a rule from **Available Rules** onto the policy bar.
9. Drag items 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 access control policies first ensures that no processing happens on responses unless the user is allowed access.

10. To remove an item from an application or resource, click **-** next to the item you want to remove.

Global unprotected resources

Global unprotected resources are resources that you specify as unprotected for all applications.

To specify a resource as unprotected for a single application, see Adding application resources.

Adding global unprotected resources

Create a new global unprotected resource in PingAccess.

About this task

> **Warning**
>
> The following steps describe how to globally make resources unprotected. Because any resource captured by the wildcard path of any entry is left unprotected for all applications, you must carefully plan these entries. To make a resource unprotected for a specific application, see Adding application resources.

Steps

1. Click **Applications**, then go to **Applications > Global Unprotected Resources**.
2. Click **+ Add Global Unprotected Resource**.
3. In the **Name** field, enter a name for the entry.

4. **Optional:** In the **Description** field, enter a description for the entry.
5. **Optional:** If you want to record access requests for this resource in the audit store, select the **Audit** check box.
6. In the **Path Pattern** field, specify the path pattern that identifies the global unprotected resource.

   This entry must start with a forward-slash (/) and can contain one or more wildcard characters (`*`), such as:

   - `/*.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** check box.
8. Click **Save**.

Editing global unprotected resources

Edit an existing global unprotected resource in PingAccess.

About this task

Change global unprotected resources within your application resources in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Global Unprotected Resources**.
2. Click to expand the global unprotected resource you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the global unprotected resource.
5. To confirm your changes, click **Save**.

Deleting global unprotected resources

Delete a global unprotected resource in PingAccess.

About this task

Remove global unprotected resources from your application resources in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Global Unprotected Resources**.

2. Click to expand the global unprotected resource you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Redirects

Redirects reroute an incoming request to another target in PingAccess.

To configure a redirect, you must map the host and port of an incoming request to that of a different target. At runtime, requests made to the source are 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

Add a new redirect in PingAccess.

About this task

Map the host and port of an incoming request to a different target.

Steps

1. Click **Applications**, then go to **Applications > Redirects**.
2. Click **+ Add Redirect**.
3. In the **Source** field, enter the source host and port that you want to redirect.
4. In the **Target** field, enter the target host and post that indicates the destination for the redirect.
5. **Optional:** To use HTTPS for the request made to the redirect target, select the **Secure Target** check box.
6. In the **Response Code** field, enter 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.
8. To confirm your changes, click **Save**.

Editing a redirect

Edit an existing redirect in PingAccess.

About this task

Change the host and port details of a redirect in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Redirects**.
2. Click to expand the redirect you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the redirect.
5. To confirm your changes, click **Save**.

Deleting a redirect

Delete an existing redirect in PingAccess.

About this task

Remove a configured redirect from PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Redirects**.
2. Click to expand the redirect you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, 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, such as `*:443` , or any host within a domain, such as `*.example.com` . If a request matches more than one virtual host, the most specific match is used.

> **Note**
>
> Prior to availability of server name indication (SNI) in Java 8, an HTTPS port could only present a single certificate. To handle multiple virtual hosts, you must 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 from the HTTPS Listeners window.

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 new virtual hosts

Create a new virtual host in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click **+ Add Virtual Host**.
3. In the **Host** field, enter the host name for the virtual host, such as myHost.com.

   You can use a wildcard (`*`) to indicate that any host name is acceptable. A wildcard host can also be specified, such as `*.example.com` .
4. In the **Port** field, enter the port number for the virtual host, such as `1234` .
5. In the **Agent Resource Cache TTL (s)** field, enter the agent resource cache TTL indicating the number of seconds the agent can cache resources for this application.

   This only applies to destination of type `Agent` .
6. To confirm your changes, click **Save**.

Configuring virtual host trusted certificate groups

Configure a virtual host trusted certificate group that can 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. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click to expand the virtual host you want to modify.
3. Click the **Pencil** icon.

   Virtual hosts that have certificate authentication configured will display the message Client Certificate Authentication in the associated bar.
4. In the **Client Certificate Authentication** field, click the **Pencil** icon.
5. From the **Trusted Certificate Group** list, select the appropriate certificate group.

   You can select an existing trusted certificate group, or use one of the following options.

Choose from:

- **No Certificate Authentication** – Does not require certificate authentication.
- **Java Trust Store** – Uses the Java Trust Store for certificate authentication.
- **Trust Any** – Allows client authentication with any certificate including self-signed certificates.

  If you use the **Trust Any** method in production, you should log client certificates in the audit log.

6. To save the trusted certificate group settings, click **Save**.
7. To confirm your changes, click **Save**.
8. Add the following two Groovy script rules to force validation of the server name indication (SNI) and client certificate chain.

   `Validate SNI`

   ```
   if(exc?.getSslData()?.getSniServerNames()?.isEmpty())
   {
     fail();
   }
   else
   {
     pass();
   }
   ```

   `Validate client certificate chain`

   ```
   if(exc?.getSslData()?.getClientCertificateChain()?.isEmpty())
   {
     fail();
   }
   else
   {
     pass();
   }
   ```

9. Apply these rules to applications that use this virtual host.

Editing virtual hosts

Edit the properties of an existing virtual host in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click to expand the virtual host you want to edit.
3. Click the **Pencil** icon.

4. Make the desired edits to the virtual host.
5. To confirm your changes, click **Save**.

Deleting virtual hosts

Delete an existing virtual host in PingAccess.

Steps

1. Click **Applications**, then go to **Applications > Virtual Hosts**.
2. Click to expand the virtual host you want to edit.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Sites

Review information for sites, site authenticators, and third-party services.

Choose from one of the following sub-sections:

- Sites operations
- Site authenticators
- Third-party services

Sites operations

Sites are the target applications, endpoints, or APIs which PingAccess protects and to which authorized client requests are forwarded.

Choose from one of the following sections:

- Adding sites
- Editing sites
- Deleting sites
- Site field descriptions

Adding sites

Add sites in PingAccess.

Steps

1. Click **Applications**, then go to **Sites > Sites**.

2. Click **+ Add Site**.
3. Complete the fields.

   For more information about the site fields, see Site field descriptions.
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 to indicate the reason that the site wasn't reachable.

Editing sites

Edit the properties of existing sites in PingAccess.

Steps

1. Click **Applications**, then go to **Sites > Sites**.
2. Click to expand the site you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the site.
5. To confirm your changes, click **Save**.

> **Note**
>
> If the target site cannot be contacted, the site is saved and a warning is displayed to indicate the reason that the site wasn't reachable.

Deleting sites

Delete an existing site in PingAccess.

Steps

1. Go to **Applications → Sites → Sites**.
2. Click to expand the site you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Site field descriptions

The following table describes the fields available for managing applications on the **Sites** page.

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 a target 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 this 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** If you clear this check box, PingAccess makes no changes to the `Host` header. This configuration is often required by target web servers to ensure that they service the HTTP request with the correct internal virtual server definition.

To configure advanced settings, expand the **Show Advanced Settings** section at the bottom of the **Site** tab.

Note

Most of these settings are optional.

Field	Required	Description
Skip Hostname Verification	No	Select this check box if the site shouldn't perform host name verification of the certificate.

Field	Required	Description
Expected Certificate Hostname	No	If you are not skipping host name 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** check box is cleared. If left blank, the certificates are verified using the target host names.
Availability Profile	No	Select an Availability profile. To create a new availability profile, click **+ Create Availability Profile**.
Load Balancing Strategy	No	If the site contains more than one target, select a load balancing strategy. To create a new load balancing strategy, click **+ Create Load Balancing Strategy**. If you specify multiple target servers for a site but don't apply a load balancing strategy, PingAccess uses the first target server in the list until it fails. Secondary target servers are only used if the first target server is not available. PingAccess uses the **Failed Retry Timeout** from the site's availability profile settings to determine when to mark the first target server as available again.
Send Token	No	If your site uses the identity information in the PingAccess token or OAuth access token, leave this check box selected to include the token in the request to the backend site. If you do not need the token information, you can clear the check box to remove the PingAccess token from the request. This excludes unnecessary information and decreases the payload size, which might improve performance.

Field	Required	Description
Maximum Connections	Yes	Enter the maximum number of HTTP persistent connections that you want PingAccess to have open and maintain for the site. The default value, `-1`, indicates unlimited connections.
Maximum Websocket Connections	Yes	If the number of WebSocket connections needs to be limited, enter a value. The default value, `-1`, indicates no limit.
Use Proxy	No	Select **Use Proxy** if requests to the site should use a configured proxy. **Note** If the node isn't 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, you should configure PingAccess to use only one target for the site. Unexpected behavior can occur if PingAccess and the proxy are both configured to perform availability handling.
Keep Alive Timeout	No	The time, in milliseconds, that a pooled connection to the site can be idle before PingAccess closes the connection and removes it from the pool. The default value, `0`, indicates no timeout.

Site authenticators

Site authenticators define the authentication mechanism that target sites require to control access.

Choose from one of the following sections:

- Adding site authenticators
- Editing site authenticators

- Deleting site authenticators

Adding site authenticators

Create a new site authenticator in PingAccess.

Steps

1. Click **Applications**, then go to **Sites > Site Authenticators**.
2. Click **+ Add Site Authenticator**.
3. In the **Name** field, enter a unique name.

   This name is shown in the **Site Authenticator** list on the **New Site** tab.

   > **Note**
   > Special characters and spaces are allowed.

4. In the **Site Authenticator** list, select the type of authentication.
5. To continue, select one of the following authentication types:

   Choose from:

   - Basic authentication site authenticators
   - Mutual TLS site authenticators
   - Token mediator site authenticators
   - SAML token mediator site authenticators

Basic authentication site authenticators

Use HTTP Basic authentication (user name:password) to authenticate a client requesting access to a site that requires basic authentication.

> **Note**
> 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 authenticators

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 trust this certificate for authentication to succeed.

> **Tip**
>
> Several steps are required for PingAccess certificate management before configuring the mutual TLS site authenticator.

Field	Description
Key Pair	The imported or 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 existing key pairs or Generating new key pairs.

Token mediator site authenticators

Token mediator site authenticators use the PingFederate Security Token Service (STS) to exchange a PingAccess token for a security token, such as a Web Access Management (WAM) token or OpenToken, that is valid at the target site.

The following table describes the fields available for managing token mediator site authenticators on the **New Site Authenticator** page.

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. For more information, see Managing Token Generators in the 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.

Field	Description
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 PingAccess 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` message.
Logged Off Cookie Value	Defines the value placed in the **Logged Off** cookie to detect an invalid or expired WAM token event.
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 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 administrative console. The administrator can then direct the token mediator site authenticator to actively return cookies to the user's browser, even when the protected site isn't doing that. Use this option to enable a seamless single sign-on (SSO) experience for users navigating from applications protected by PingAccess to applications protected by a third-party WAM 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 when you're using non-HTTP methods, such as making calls through JavaScript. For example, referencing `document.cookie` in a JavaScript call.
Secure Cookie	Indicate whether the generated logged-in cookie must be sent using only HTTPS connections.

Field	Description
Token Processor ID	Defines the instance name of a token processor that you want to use. The token processor is configured in PingFederate. Specify this value if more than one instance of either the JSON Web Token (JWT) 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.

SAML token mediator site authenticators

Security Assertion Markup Language (SAML) token mediator site authenticators use the PingFederate Security Token Service (STS) to exchange a PingAccess token for a SAML token that is valid at the target site.

The following table describes the fields available for managing SAML token mediator site authenticators on the **New Site Authenticator** page.

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. For more information, see Managing token generators in the 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 In Header Name	Defines the header name containing the token that the target site is expecting. You must enter a valid header name per RFC 7230. **Note** You can set both a **Logged In Cookie Name** and a **Logged In Header Name** for a SAML token mediator site authenticator, or you can just pick one, but you must fill out at least one of these two fields.

Field	Description
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 PingAccess token is still valid, PingAccess re-obtains a valid SAML 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` message.
Logged Off Cookie Value	Defines the value placed in the **Logged Off** cookie to detect an invalid or expired SAML token event.

Advanced Settings

To configure advanced settings on a SAML token mediator site authenticator, expand the **Show Advanced Settings** section at the bottom of the **New Site Authenticator** page. These settings are optional.

Field	Description
Token Processor ID	Defines the instance name of a token processor that you want to use. The token processor is configured in PingFederate. Specify this value if more than one instance of either the JSON Web Token (JWT) 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.

Editing site authenticators

Edit the properties of an existing site authenticator.

Steps

1. Click **Applications**, then go to **Sites > Site Authenticators**.
2. Click to expand the site authenticator you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the site authenticator.
5. To confirm your changes, click **Save**.

Deleting site authenticators

Delete existing site authenticators in PingAccess.

About this task

> **Important**
>
> You cannot delete site authenticators if they're associated with a site.

Steps

1. Click **Applications**, then go to **Sites > Site Authenticators**.
2. Click to expand the site authenticator you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

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. Choose from one of the following topics:

- Adding third-party services
- Editing third-party services
- Deleting third-party services
- Third-party service field descriptions

Adding third-party services

Add new third-party services in PingAccess.

Steps

1. Click **Applications**, then go to **Sites > Third Party Services**.
2. Click **+ Add Third-Party Service**.
3. Complete the fields.

   For information about completing the fields, see Third-party service field descriptions.
4. Click **Save**.

Editing third-party services

Edit the properties of existing third-party services in PingAccess.

Steps

1. Click **Applications**, then go to **Sites > Third Party Services**.
2. Click to expand the third-party service you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the third-party service.
5. To confirm your changes, click **Save**.

Deleting third-party services

Delete existing third-party services in PingAccess.

Steps

1. Click **Applications**, then go to **Sites > Third Party Services**.
2. Click to expand the third-party service you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Third-party service field descriptions

The following table describes the fields available for managing the applications listed in **Sites → Third-Party Services** in PingAccess.

Field	Required	Description
Name	Yes	Specify a name that identifies the third-party service.
Targets	Yes	Specify one or more host name:port pairs used to reach the third-party service.
Secure	No	Indicate whether or not the target is expecting a secure connection.

To configure advanced settings, expand the **Show Advanced Settings** section at the bottom of the **Third-Party Service** tab.

> **Note**
>
> Most of these settings are optional.

Field	Required	Description
Host Value	No	When you specify a **Host Value**, requests to a third-party service use this value as the host header field value regardless of which target is used.
Skip Hostname Verification	No	For secure connections, select this check box to indicate that the third-party service shouldn't perform host name verification of the certificate.
Expected Certificate Hostname	No	For secure connections, enter the name of the host expected in the certificate when host name verification is enabled.
Availability Profile	Yes	Indicate the availability profile to use. To create a new availability profile, click **+ Create Availability Profile**.
Load Balancing Strategy	No	Select the load-balancing strategy to use if more than one target is defined. If you specify multiple target servers for a third-party service but don't apply a load balancing strategy, PingAccess uses the first target server in the list until it fails. Secondary target servers are only used if the first target server is not available. PingAccess uses the **Failed Retry Timeout** from the service's availability profile settings to determine when to mark the first target server as available again.
Maximum Connections	Yes	Indicates the maximum number of HTTP-persistent connections PingAccess will open and maintain for the service. The default value, `-1`, indicates unlimited connections.
Use Proxy	No	Indicates that requests to the site should use a configured proxy.

Agents

You can manage agents in PingAccess. Agents are web server plugins 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 through the PingAccess Agent Protocol (PAAP), which defines the possible interactions between agents and the policy server. Agents have identifying names and a shared secret to authenticate with the policy server. Agents do not need to be unique. There can be multiple agents using the same name and secret, and they are all treated equally by the policy server. This is useful in complex deployments where unique agents are difficult to manage. Agents can be assigned as the destination for one or more applications by name.

Assigning agent listener key pairs

Before you create an agent, 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 needs to include the host or hosts the agent will use to contact the PingAccess policy server.

2. Click **Security**, then go to **Key Pairs > Key Pairs**.
3. Click the **Pencil** icon.
4. Click **Assign HTTPS Listener** for the key pair.
5. Select the **Agent** check box.
6. Click **Save**.

> **Note**
>
> If the environment is clustered, check the `pingaccess.log` file on each engine to ensure replication completed before restarting each engine.

Adding agents

Create new agents in PingAccess.

Steps

1. In the PingAccess console, go to **Applications > Agents**.
2. Click **+ Add Agent**.
3. Complete the fields.

   You can find more information about the fields in Agent field descriptions.

4. To configure advanced settings, click **Show Advanced**.
5. To save the configuration and download the `<agent-name>_agent.properties` file for use with the PingAccess agent, click **Save & Download**.

   > **Note**
   >
   > The shared secret is generated by the PingAccess server and identified on this page with a timestamp. You can delete existing secrets by clicking **Remove** 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.

   PingAccess can generate additional agent `agent.properties` files containing the specified information that can configure the agent plugin. Existing configurations can also be downloaded again if necessary.

Editing agents

Edit existing agents in PingAccess.

Steps

1. In the PingAccess console, go to **Applications > Agents**.
2. Click to expand an existing agent you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits.
    1. To download the shared secret, click **Download**.
    2. To remove the shared secret, click **Remove**.
5. Click **Save & Download**.

Deleting agents

Delete existing agents in PingAccess.

Steps

1. In the PingAccess console, go to **Applications > Agents**.
2. Click to expand an existing agent you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

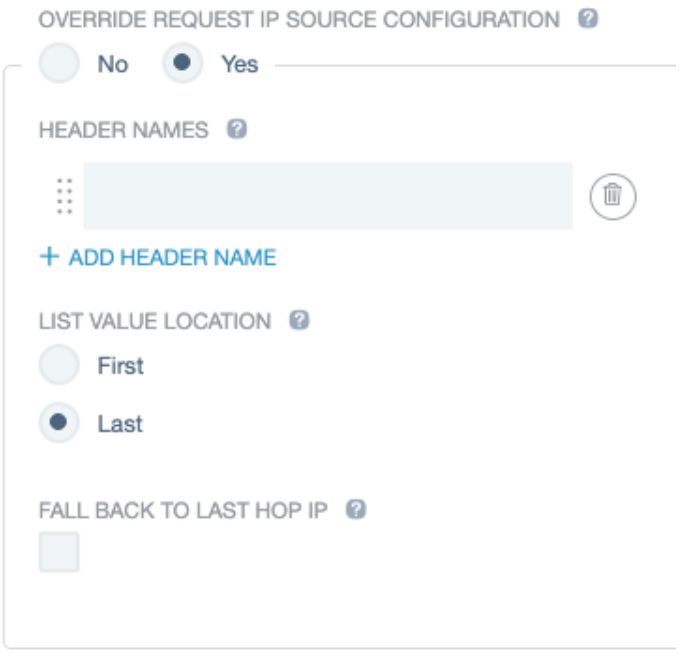
Agent field descriptions

The following table describes the fields available for managing applications in the **Agents** window.

Field	Required	Description
Name	Yes	Enter a unique alphanumeric name for the agent, up to 64 characters.
Description	No	Enter a unique description to identify the agent's purpose.

Field	Required	Description
PingAccess Host	Yes	In the **PingAccess Host** fields, enter the **Hostname** and **Port** of the PingAccess server where the agent should send requests. **Note** The PingAccess **Hostname** and **Port** might not be the actual host and port that policy server is listening to, depending on the network routing configuration and network elements, such as reverse proxies and load balancers. The PingAccess **Host** and **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 **Port** values might point to the load balancer rather than directly to an engine host. This configuration provides 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 can be added using the application programming interface (API).
Agent Trusted Certificate	Yes	Select a certificate to export in the `<agent-name>_agent.properties` file. The agent uses this certificate to communicate with the PingAccess engine using Secure Sockets Layer (SSL)/TLS. PingAccess determines the certificates that are available to select in this list based on the certificates that have been imported into PingAccess. If the certificate you want to use isn't available, you must import it into the system. Learn more in Importing certificates. **Note** You must specify the certificate authority (CA) root certificate if the agent listener presents a CA-signed certificate chain. The default value is `C=US, O=Ping Identity, CN=localhost (Generated: AGENT)`.

To configure advanced settings, click **Show Advanced**.

Field	Required	Description
Override Request IP Source Configuration	No	When enabled, the configuration you provide in the following fields overrides the default IP source settings (defined in **Settings > HTTP Requests > IP Source**) for this agent instance.  To configure the agent to use different Internet Protocol (IP) source information: 1. In the **Override Request IP Source Configuration** section, select **Yes**. 2. In the **Header Names** section, click **+ Add Header Name** and enter a header name which identifies the source IP address. To enter multiple header names, click **+ Add Header Name** for each header. 3. If you include more than one value in the **Header Names** section, in the **List Value Location** section, select whether the first or last value in the list is the source address. The default value is **Last**. 4. Select the **Fall Back to Last Hop IP** checkbox to use the last hop IP address as the source address if none of the configured header names are found. **Note** If you don't select this option and none of the listed header names are found, access is denied and a `Forbidden` result is returned.

Field	Required	Description
Override Unknown Resource Configuration	No	When enabled, the configuration you provide in the following fields overrides the default unknown resource settings (defined in **Settings > Access > Unknown Resources**) for this agent instance.  To configure the agent to handle unknown resource requests differently: 1. In the **Override Unknown Resource Configuration** section, select **Yes**. 2. In the **Mode** field, select one of the following options to specify how requests for unknown resources should be handled: ***Deny*** Agent requests for unknown resources generate an error response. ***Passthrough*** Agent requests for unknown resources are allowed to pass through unfiltered.

Field	Required	Description
Require Token Authentication	No	Select this checkbox to require the PingAccess agent to use bearer token authentication when making requests to the PingAccess engine nodes. **Tip** By default, agents send both the shared secret and the bearer token if you select this checkbox. To prevent an agent from sending the shared secret, remove the `agent.engine.configuration.shared.secret` property from the `agent.properties` file you download. You can find more information about setting up bearer token authentication in Configuring PingAccess agents to use bearer token authentication. **Important** Only the PingAccess agent for Apache (RHEL), the PingAccess agent for Apache (SLES), and the PingAccess agent for NGINX have been updated to support bearer token authentication currently. Learn more in Agent SDK for C 3.0 (April 2025). Keep this checkbox cleared until compatibility is added for the type of agent you're using, and you've updated all agents to the supported version.
Max Retries	Yes	Enter a number specifying how many times an agent should try contacting a PingAccess server before considering it unavailable. The default value is `2`.
Failed Retry Timeout	Yes	Enter a number, in seconds, specifying how long an agent should wait before trying to establish a connection to a failed PingAccess server again. The default value is `60`.

Configuring PingAccess agents to use bearer token authentication

Authenticate PingAccess agents to the engine nodes with bearer token authentication in addition to, or instead of, a shared secret.

About this task

When you enable bearer token authentication, PingAccess engine nodes:

- Require PingAccess agents to send a signed JSON Web Token (JWT) with all HTTP requests.
- Verify that the JWT was signed by the expected key.

- Log debug messages to confirm that the token was received and validated as expected.

> **Important**
>
> Only the PingAccess agent for Apache (RHEL), the PingAccess agent for Apache (SLES), and the PingAccess agent for NGINX have been updated to support bearer token authentication currently, but the `agent.properties` file includes a private key as of PingAccess 8.2.
> Compatibility for the Apache (Windows) and IIS agents will be added in a future release. You can download and use the updated `agent.properties` file normally in the meantime. Complete the **Configuring incompatible PingAccess agents to test bearer token authentication** procedure to do so.

Configuring compatible PingAccess agents to use bearer token authentication

About this task

Complete this procedure to configure either the PingAccess agent for Apache (RHEL), the PingAccess agent for Apache (SLES), or the PingAccess agent for NGINX 3.0 to use bearer token authentication.

Steps

1. In the PingAccess admin console, go to **Applications > Agents** and open the agent configuration that you want to update.
2. To prompt PingAccess to add the private key into the `agent.properties` file, select the **Require Token Authentication** checkbox.

   > **Tip**
   >
   > If you clear this checkbox later, you don't need to generate a new `agent.properties` file to update the shared secret. The PingAccess agent will continue to use both the shared secret and the private key from the active `agent.properties` file if you haven't removed them from the file.

3. Download a new `agent.properties` file for the agent as shown in Adding agents.

   In PingAccess 8.2 and later, the PingAccess server generates a public key and private key in addition to the shared secret. You can find the public key on this page, identified with a timestamp. The updated `agent.properties` file contains the expected private key.

   > **Note**
   >
   > To rotate keys, generate a new `agent.properties` file, then remove the old file and public key.

4. Configure the agent with the updated `agent.properties` file.

   > **Important**
   >
   > When the private key is present in the `agent.properties` file, it will genereate a unique JWT for authentication with every request to the PingAccess server. The JWT expires after 2 minutes, so you must ensure you synchronize the agent and the PingAccess server's clocks.

5. Repeat steps 1 - 5 for all configured agents.

Configuring incompatible PingAccess agents to test bearer token authentication

About this task

Complete this procedure to configure either the PingAccess agent for Apache (Windows) or the PingAccess agent for IIS to test bearer token authentication.

Steps

1. In the PingAccess admin console, go to **Applications > Agents** and open the agent configuration that you want to update.
2. To prompt PingAccess to add the private key into the `agent.properties` file, select the **Require Token Authentication** checkbox.

   > **Tip**
   >
   > If you clear this checkbox later, you don't need to generate a new `agent.properties` file to update the shared secret. The PingAccess agent will continue to use both the shared secret and the private key from the active `agent.properties` file if you haven't removed them from the file.

3. Download a new `agent.properties` file for the agent as shown in Adding agents.

   In PingAccess 8.2 and later, the PingAccess server generates a public key and private key in addition to the shared secret. You can find the public key on this page, identified with a timestamp. The updated `agent.properties` file contains the expected private key.

   > **Note**
   >
   > To rotate keys, generate a new `agent.properties` file, then remove the old file and public key.

4. Clear the **Require Token Authentication** checkbox.

   > **Important**
   >
   > After downloading the new `agent.properties` file, leave the **Require Token Authentication** checkbox cleared until agent compatibility is added and this agent has been updated to the supported version.

5. (Optional) To confirm that shared secret authentication still works as expected, configure the agent with the updated `agent.properties` file.

   > **Important**
   >
   > After the agents have been updated to support bearer token authentication, make sure to download the latest version of the agents and configure them with the updated `agent.properties` files.
   > Following that, you can select the **Require Token Authentication** checkbox to require the configured PingAccess agent to use bearer token authentication in addition to the shared secret when making requests to the PingAccess engine nodes.

6. Repeat steps 1 - 5 for all configured agents.

Sideband Clients

Use sideband clients to query the PingAccess Sideband API for authorization and request and response modification for API gateways. This lets PingAccess supply access decisions to gateway tools.

This section covers the following:

- Adding sideband clients
- Editing sideband clients
- Deleting sideband clients

Adding sideband clients

Create new sideband clients that will query PingAccess for access decisions.

Before you begin

Verify that the `sideband.http.enabled` property is set to `true` in the PingAccess configuration file. You can find more information in the Configuration file reference.

Steps

1. Click **Applications**, then go to **Sideband Clients**.
2. Click **+ Add Sideband Clients**.
3. In the **Name** field, enter a unique name for the sideband client.
4. (Optional) In the **Description** field, enter a description for the sideband client.
5. In the **Secrets** section, add a secret:

   Result:

   The **New Secret** page opens.
6. On the **New Secret** page, click **Copy** to copy the new secret to your clipboard.

   Save the secret in a secure location.

   > **Note**
   >
   > When you use the secret, replace the `<client name>` placeholder value with the **Name** you configured in step 3.
7. Click **Done**.
8. (Optional) In the **Header Name** field, enter a unique, descriptive name to make it easier to tell your configured secrets apart.

> **Note**
>
> Configure your sideband integration to send a request header using the configured **Header Name** as the header field value.

9. (Optional) Repeat steps 5 - 8 to create additional secrets.
10. Click **Save**.

Editing sideband clients

Edit existing sideband clients in PingAccess to update their characteristics.

Steps

1. Click **Applications**, then go to **Sideband Clients**.
2. Click the **Expand** icon to expand an existing sideband client you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits.
5. Click **Save**.

Deleting sideband clients

Delete existing sideband clients in PingAccess to remove a client's access.

Steps

1. Click **Applications**, then go to **Sideband Clients**.
2. Click to expand an existing sideband client you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Access header

The **Access** header contains options related to access control, such as rules, web sessions, and token validation.

The **Access** header contains these menu options:

- Rules
- Authentication
- Identity mappings
- Web sessions

- Token validation
- Unknown resources
- Managing risk policies

Rules

The PingAccess policy manager contains controls for adding and managing rules. Use rules to 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 an interface in the PingAccess administrative console where you can create rules, rule sets, and rule set groups, and apply them to applications and application resources. Policies are the rules, rule sets, or groups of rule sets applied to a specific 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 within the policy to decide whether the client can access the application resource and whether any additional actions need to occur before granting that access.

For information on how to assign rules, rule sets, and rule set groups, see applying rules to applications and resources.

Rule types

Access control rules

Access control rules can restrict access in a number of ways. For example, an access control rule might:

- Test user attributes (for more information, see OAuth attribute rules)
- Check the time of day the request was made at (for more information, see time range rules
- Request Internet Protocol (IP) addresses (for more information, see network range rules)
- Test OAuth access token scopes (for more information, see OAuth scope rules)

> **Tip**
>
> Ensure that any headers used in access control rules, such as the `X-Forwarded-For` header that network range rules use, are sanitized and managed exclusively by inline infrastructure that users must be routed through before reaching PingAccess and the protected applications.

Processing rules

Processing rules can perform request processing. For example, a processing rule might:

- Modify headers (for more information, see rewrite response header rules)
- Rewrite URLs (for more information, see rewrite URL rules)

Processing order

Access control rules are applied before processing rules. For each type of rule, the rules are applied in the order configured in the policy manager. All rules are evaluated after identity mappings are, so that the 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, PingAccess applies the rules in the following order:

1. Application access control rules
2. Resource access control rules
3. Resource processing rules
4. Application processing rules

Agent deployments

The following rules aren't available for agent deployments:

- Adding PingAuthorize response filtering rules
- Rewrite content rules
- Rewrite cookie domain rules
- Rewrite cookie path rules
- Rewrite response header rules
- Rewrite URL rules

Rule management

You can create and manage rules to control access to your web apps and APIs. PingAccess supports a variety of rule types, and the procedures for rule creation are different for each rule type.

Choose from one of the following topics:

- Creating access control rules
- Creating processing rules
- Editing rules
- Deleting rules

Creating access control rules

Create access control rules to control how and when users can access sites and APIs.

For more information about access control rules in PingAccess, see the following topics:

- Adding an authentication requirements rule
- Adding Groovy script rules
- Adding HTTP request header rules
- Adding HTTP request parameter rules

- Adding network range rules
- Adding OAuth attribute rules
- Adding OAuth client rules
- Adding OAuth Groovy script rules
- Adding OAuth scope rules
- Adding one-time authorization rules
- Adding PingAuthorize access control rules
- Adding PingAuthorize policy decision access control rules
- Adding rate limiting rules
- Adding redirect rules
- Adding rejection rules
- Adding time range rules
- Adding web session attribute rules
- Adding web session scope rules
- Adding WebSocket handshake rules

Adding an authentication requirements rule

Add an authentication requirements rule in PingAccess to limit access to resources or applications protected by PingAccess based on the access control rule (ACR) values returned by the PingFederate request AuthN context authentication selector.

Before you begin

Verify that you have:

- A PingFederate configuration that uses the `Requested AuthN Context Authentication Selector`
- A configured authentication list

About this task

An authentication requirements rule 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.

This rule also allows for configurations that require more secure authentication methods, such as multi-factor authentication (MFA). For example, a website might allow a user to authenticate and view personal data using only a user name 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.

> **Important**
>
> To ensure that step-up authentication is triggered, this rule should always be positioned first in a list of rules, rule sets, or rule set groups, regardless of whether the criteria is `Any` or `All`.
> PingAccess uses rules to trigger different authentication paths in PingFederate. If the authentication requirements rule isn't the first item in a list, then it isn't sent to PingFederate in the initial request.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, 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. **Optional:** In the **Max Age (M)** field, enter a maximum time since the last authentication. If the user's session has not authenticated in this timeframe, the user is prompted to reauthenticate.

   A value of -1 indicates no maximum age.
8. Click **Save**.

Adding Groovy script rules

Add a Groovy script rule to provide advanced rule logic that extends PingAccess rule development beyond the capabilities of the packaged Rules.

About this task

> **Note**
>
> Through Groovy scripts, PingAccess administrators can perform sensitive operations that might affect system behavior and security. 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 application programming interface (API), respectively, in the rules list.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Groovy Script (for Web App)**.
5. In the **Groovy Script** field, enter the Groovy script to use for rule evaluation.

   Example:

   To validate that an HTTP request does not come from Internet Explorer, you might include the `requestHeaderDoesntContain("User-Agent", "InternetExplorer")` PingAccess matcher in your Groovy script.

   Groovy script rules must end execution with a matcher instance. For more information, see Matcher usage reference.
6. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

   Choose from:

   - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
   - If you select **Basic**, 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. If you select **Basic**, provide the following:

     1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

        The default is `403`.
     2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

        The default is `Forbidden`.
     3. In the **Error Response Template File** field, enter 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.
     4. From the **Error Response Content Type** list, select the type of content for the error response.

        This lets the client properly display the response.
7. Click **Save**.

Adding HTTP request header rules

Add an HTTP request header rule to examine a request and determine 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

If more than one **Field** and **Value** pair is listed, then all conditions must match in order for the rule to succeed.

Steps

1. Click **Access**, then go to **Rules** > **Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **HTTP Request Header**.
5. In the **Field** column, in the **Header** field, enter a header name you want to match to grant or not grant the client access.
6. In the **Value** field, enter a 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 in the **Value** field, or add a wildcard after the hostname (`host*` or `host:*`) to match the HTTP request.
7. If additional header pairs are needed, click **Add Row** to add an additional row, then repeat steps 5-6.
8. If the values should be an exact match to the value case, select the **Case Sensitive** check box.
9. If access is not allowed when a match is found, select the **Negate** check box.

   > **Note**
   >
   > 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.
10. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

    Choose from:

    - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.

- If you select **Basic**, 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. If you select **Basic**, provide the following:

    1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

       The default is `403`.

    2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

       The default is `Forbidden`.

    3. In the **Error Response Template File** field, enter 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.

    4. From the **Error Response Content Type** list, select the type of content for the error response.

       This lets the client properly display the response.

11. Click **Save**.

Adding HTTP request parameter rules

Add an HTTP request parameter rule to examine a request and determine whether to grant access to a requested resource based on a match found in specified form parameters of the HTTP request.

About this task

Add an HTTP request parameter rule determines if the parameters are passed as part of the Uniform Resource Locator (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 for the rule to succeed.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.

4. From the **Type** list, select **HTTP Request Parameter**.

5. In the **Field** column, in the **Parameter** field, enter a parameter name you want to match to grant or not grant the client access.
6. In the **Value** field, enter a value for the parameter 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 convert to `v1%20v2` before the search is performed.

7. If additional parameters pairs are needed, click **Add Row** to add an additional row, then repeat steps 5-6.
8. If the values should be an exact match to the value case, select the **Case Sensitive** check box.
9. If access is not allowed when a match is found, select the **Negate** check box.

   > **Note**
   >
   > 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.

10. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

    Choose from:

    - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
    - If you select **Basic**, 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. If you select **Basic**, provide the following:

      1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

         The default is `403` .

      2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

         The default is `Forbidden` .

      3. In the **Error Response Template File** field, enter 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.

      4. From the **Error Response Content Type** list, select the type of content for the error response.

         This lets the client properly display the response.

11. Click **Save**.

Adding network range rules

Add a network range rule to examine a request and determine 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. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **Network Range**.
5. In the **Network Range** field, enter a network range value, such as `127.0.0.1/8`.

   PingAccess supports IPv4 addresses.
6. Select **Negate** if when a match is found, access is not allowed.
7. If you want to override source address handling defined in the **HTTP Requests** configuration, click **Show Advanced Settings** and perform the following steps:
   1. Click **Override Request IP Source Configuration**.
   2. In the **Headers** field, enter the headers used to define the source IP address to use.
   3. Select the **Header Value Location** to use when multiple addresses are present in the specified header.

      Valid values are `Last` (the default) and `First`.
   4. Click **Fall Back to Last Hop IP** 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.
   5. To configure rejection handling, select a rejection handling method:

      Choose from:

      - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
      - If you select **Basic**, 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. If you select **Basic**, provide the following:
        1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

           The default is `403`.

2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

   The default is `Forbidden`.

3. In the **Error Response Template File** field, enter 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.

4. In the **Error Response Content Type** list, select the type of content for the error response.

   This lets the client properly display the response.

8. Click **Save**.

Adding OAuth attribute rules

Add an OAuth attribute rule to examine a request and determine 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 rule.

Steps

1. Click **Access**, then go to **Rules** > **Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.

4. From the **Type** list, select **OAuth Attribute**.
5. From the **Attribute Name** list, select the attribute name you want to match to an attribute associated with an OAuth access token.
6. In the **Attribute Value** field, enter the value to match.

   > **Note**
   >
   > The attribute values come from the contract in your OAuth access token manager in PingFederate. For more information, see Defining access token attribute contract.

7. Add additional rows of attribute name and value pairs as needed.

   > **Note**
   >
   > If multiple rows are included here, all conditions must match for the rule to match.

8. Select **Negate** if, when a match is found, access is not allowed.

> **Note**
>
> Verify what you enter for the attribute. If you enter an attribute that does not exist, such as if the attribute is misspelled, and you select **Negate**, the rule will always succeed.

9. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

   Choose from:

   - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
   - If you select **Basic**, 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. If you select **Basic**, provide the following:

     1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

        The default is `403`.

     2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

        The default is `Forbidden`.

     3. In the **Error Response Template File** field, enter 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.

     4. From the **Error Response Content Type** list, select the type of content for the error response.

        This lets the client properly display the response.

10. Click **Save**.

Adding OAuth client rules

Add an OAuth client rule to restrict access to API applications based on one or more OAuth client IDs.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.

4. From the **Type** list, select **OAuth Client**.
5. In the **Client IDs** section, enter one or more Client IDs that allow access. To add additional fields, click **+ New Value**.

6. **Optional:** If you want to configure rejection handling, click **Show Advanced Settings**, and then from the **Rejection Handler** list, select an existing rejection handler that defines whether to display an error template or redirect to a URL.

   > **Note**
   >
   > You can include information about missing Client IDs in the rejection response using the `$info` variable. For example, if you are using the Default application programming interface (API) rejection handler, you could edit the `<PA_HOME>/conf/template/oauth.error.json` file and change this line: `{"$Encode.forJavaScriptSource($header)":""}`
   > to
   > `{"$Encode.forJavaScriptSource($header)":"#if($info)$Encode.forJavaScriptSource($info)#end"}`

7. Click **Save**.

Adding OAuth Groovy script rules

Add an OAuth Groovy script rule to determine 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

Adding an OAuth Groovy script rule allows you to create more sophisticated OAuth scope and OAuth attribute value rules for application programming interface (API) applications.

> **Note**
>
> 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 list.

Steps

1. Click **Access**, then go to **Rules** > **Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **OAuth Groovy Script (for API)**.
5. In the **Groovy Script** field, enter the Groovy script to use for rule evaluation.

   Example:

   To create an OAuth scope rule that matches more than one scope, you might include the `hasScopes("access","portfolio")` matcher in your Groovy script.

> **Note**
>
> Groovy script rules must end execution with a matcher instance. For more information, see Matcher usage reference.

6. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

   Choose from:

   - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
   - If you select **Basic**, 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. If you select **Basic**, provide the following:

     1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

        The default is `403`.

     2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

        The default is `Forbidden`.

     3. In the **Error Response Template File** field, enter 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.

     4. From the **Error Response Content Type** list, select the type of content for the error response.

        This lets the client properly display the response.

7. Click **Save**.

Adding OAuth scope rules

Add an OAuth scope rule to examine the contents of the PingFederate validation response and determine whether to grant access to a backend target site based on a match found between the scopes of the validation response and scope specified in the rule.

About this task

For example, a resource might require that the OAuth access token contain the scope `superuser`.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

Special characters and spaces are allowed.

4. From the **Type** list, select **OAuth Scope**.
5. From the **Scope** list, select the scope you want to match to values returned from the access token.

> **Note**
>
> 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. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

   Choose from:

   - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
   - If you select **Basic**, 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. If you select **Basic**, provide the following:

     1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

        The default is `403`.

     2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

        The default is `Forbidden`.

     3. In the **Error Response Template File** field, enter 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.

     4. From the **Error Response Content Type** list, select the type of content for the error response.

        This lets the client properly display the response.

8. Click **Save**.

Adding one-time authorization rules

Add a one-time authorization rule to let the user obtain authorization for a mobile app or single-page application using the Client-Initiated Back-channel Authentication (CIBA) specification.

Before you begin

You must have a configured token provider and an OAuth client with the client-initiated backchannel authentication (CIBA) grant type enabled.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **One-Time Authorization**.
5. In the **Client ID** field, enter the Client ID of the OAuth client.
6. Select a **Client Credentials Type**, then provide the information required for the selected credential type.

   Choose from:

   - **Secret** – In the **Client Secret** field, enter the secret used by the OAuth client to authenticate to the authorization server.
   - **Mutual TLS** – From the **Mutual TLS** list, select a configured **Key Pair** to use for Mutual TLS client authentication.
   - **Private Key JWT** – Select this option to use Private Key JSON web token (JWT). No additional information is required.
7. From the **Login Hint Request Attribute** list, select an attribute.

   When a user authenticates, the value of this attribute is included in the call to the token provider. This attribute value can identify the user.
8. **Optional:** In the **Scopes** field, enter or select a scope to request from the token provider. The `openid` scope is automatically requested.
   1. **Optional:** Click **+ New Value** to add additional fields.
9. **Optional:** Click **Show Advanced** to configure advanced options:
   1. **Optional:** In the **Requested Expiry (S)** field, enter the transaction lifetime in seconds.

      If not specified, the value defined in the CIBA request policy is used.
   2. **Optional:** From the **Timeout Rejection Handler** list, select the handler to use for an expired request.
   3. **Optional:** From the **Deny Rejection Handler** list, select the handler to use for a denied request.
10. Click **Save**.

Adding PingAuthorize access control rules

Add an access control rule to contact PingAuthorize or PingOne Authorize for access information.

> **Note**
>
> To use this rule with PingOne Authorize, make sure that the **Include Identity Attributes** checkbox is selected in step 7 of this procedure.

Before you begin

Create a third-party service with PingAuthorize configured as the target. Learn more in Adding third-party services.

About this task

An access control rule can grant or deny access and can modify the request, based on the response from the PingAuthorize request application programming interface (API).

> **Important**
>
> The PingAuthorize sideband API cannot accept gzipped data from upstream server responses. Ensure that upstream server requests add or replace the `Accept-Encoding` header with `Accept-Encoding: identity` to prevent the upstream server from sending compressed responses.

PingAuthorize access control rules are available for gateway, sideband, and agent deployments.

> **Note**
>
> In agent deployments, PingAuthorize access control rules have the following limitations:
>
> - Agents cannot provide the request body to PingAuthorize.
> - Agent caching is disabled for resources or applications that use the PingAuthorize access control rule.

To add a PingAuthorize access control rule:

Steps

1. In the PingAccess administrative console, click **Access** and go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name of up to 64 characters.

   Special characters and spaces are allowed.
4. In the **Type** list, select **PingAuthorize Access Control**.
5. In the **Third Party Service** list, select your PingAuthorize service.
6. In the **Shared Secret** field, enter the shared secret from PingAuthorize.
7. (Optional) To include access token data in the request to PingAuthorize, select the **Include Identity Attributes** checkbox.

   This option is selected by default.

   > **Note**
   >
   > If you're using PingOne Authorize, this checkbox must be selected. PingOne Authorize requires identity attributes.

8. (Optional) To include the HTTP request body in the HTTP request data sent to PingAuthorize, select the **Include Request Body** checkbox.

   If PingAuthorize needs the request body for an access decision, make sure that this checkbox is selected. Otherwise, clearing the checkbox could improve performance.

   This option is selected by default.

9. (Optional) To configure advanced options, click **Show Advanced**:

   1. (Optional) In the **Sideband Endpoint** field, enter the sideband API endpoint location.
   2. (Optional) In the **Shared secret header name** field, enter a header in which to send the shared secret.
   3. (Optional) In the **Additional Request Headers** section, enter a **Header Name** and **Header Value** for any additional headers that you want to include in the request to PingAuthorize. Click **+ Add Row** to add other headers as necessary.

      PingAuthorize can use the additional headers to determine the policy set that's most relevant to the request context.

      If an additional header that you configured appears in a user request, PingAccess replaces the original request header and its corresponding values with the **Header Value** that you configured. If you leave the **Header Value** field blank, PingAccess removes this header from the request to PingAuthorize.

      If the **Header Value** contains the substrings `"${APPLICATION_NAME}"` or `"${RESOURCE_NAME}"`, PingAccess replaces those strings with the name of the requested application or resource as defined in PingAccess.

10. Click **Save**.

Adding PingAuthorize policy decision access control rules

Add a policy decision access control rule to use the Policy Decision Endpoint to contact PingAuthorize for access information.

> **Note**
>
> This rule allows for more control over fine-grain authorization decisions than the PingAuthorize access control rule, but isn't compatible with PingOne Authorize.

Before you begin

- Create a third-party service with PingAuthorize configured as the target. Learn more in Adding third-party services.
- Make the JSON PDP API available. Learn more in About the Authorization Policy Decision APIs.

Limitations:

- Currently, there isn't an out-of-the-box way to perform token validation for PDP calls. If you want to use token validation instead of shared secrets, learn more in Setting up the token validation workaround.

About this task

Use a policy decision access control rule to:

- Set the domain, service, and action.
- Provide additional attributes to be sent with the request. This includes static attributes, the request method and request URI, and attributes from the token provider.
- Retrieve information that PingAuthorize returns as a statement and set it as a header.

To add a PingAuthorize policy decision access control rule:

Steps

1. Click **Access** and then go to **Rules > Rules**.
2. Click **Add Rule**.
3. In the **Name** field, enter a unique name of up to 64 characters.

   Special characters and spaces are allowed.
4. In the **Type** list, select **PingAuthorize Policy Decision Access Control**.
5. In the **Third Party Service** list, select your PingAuthorize service.
6. In the **Authorization Policy Decision Shared Secret** field, enter the shared secret from PingAuthorize.
7. In the **Domain** field, enter the organizational structure in the trust framework to use.
8. In the **Service** field, enter either the service PingAuthorize protects or the data source for the policy decision.
9. In the **Action** field, enter the action that the authorization request might ask to perform on a specific resource.
10. To send the access token to the PingAuthorize service, select the **Send Web Session Access Token as PingAuthorize Request Attribute** checkbox.
11. In the **Mapped Payload Attributes** section, enter any attributes to send to the PingAuthorize service.
    1. Click **+ Add Row** to create a mapping.
    2. In the **Attribute Name** field, enter the name of the attribute.
    3. In the **Attribute Value** list, select the value to which you want to map the attribute.
12. In the **Static Payload Attributes** section, enter any static attributes to send to the PingAuthorize service.
    1. Click **+ Add Row** to create a mapping.
    2. In the **Attribute Name** field, enter the name of the attribute.
    3. In the **Attribute Value** field, enter the value to which you want to map the attribute.
13. In the **Mapped Request Object Payload Attributes** section, enter any request object attributes to send to the PingAuthorize service.
    1. Click **+ Add Row** to create a mapping.
    2. In the **Attribute Name** field, enter the name of the attribute.

3. In the **Attribute Value** list, select the value to which you want to map the attribute.

14. In the **Mapping Statement Response to Headers** section, enter a JSON pointer to identify any data in the returned statement from PingAuthorize that can be mapped to a header for the response.
    1. Click **+ Add Row** to create a mapping.
    2. In the **Header Name** field, enter the name of the header to which you want to map the data.
    3. In the **Response JSON Pointer** field, enter a JSON pointer that indicates the data in the PingAuthorize statement that you want to map.
15. (Optional) To configure advanced settings, click **Show Advanced Settings**.
    1. In the **Shared Secret Header Name** field, enter the header in which to send the shared secret.
    2. In the **Rejection Handler** list, select a rejection handler.
16. Click **Save**.

Setting up the token validation workaround

Shared secrets are available out of the box with the PingAuthorize policy decision access control rule, but token validation is not. You can perform the following workaround to set up token validation.

> **Note**
>
> This procedure is optional.

About this task

To set up token validation, you'll build the call to the introspection endpoint manually and create a policy to check if the token is active.

There are three main steps in this workaround:

1. Creating an HTTP service for the PingFederate introspection endpoint
2. Creating attributes for the `client_id`, `client_secret`, and active token
3. Creating a policy to validate the access token

To create an HTTP service for the PingFederate introspection endpoint:

Steps

1. In the PingAuthorize administrative console, go to **Trust Framework > Services > PDP** and create a service named `PingFederate Introspection`.
2. In the **Service Type** list, select **HTTP**.
3. In the **HTTP Settings** section:
    1. In the **URL** field, enter the PingFederate introspection endpoint.
    2. In the **HTTP Method** list, select `POST`.
    3. In the **Body** field, enter `token={{HttpRequest.AccessToken.access_token}}`

4. In the **Authentication** list, select **Basic**.
5. In the **Username** list, select the PingFederate client option.
6. In the **Password** list, select the PingFederate client option.

4. In the **Value Processors** section, add a new JSON path processor:
    1. In the **Processor** list, select **JSON** and enter `active` in the field.
    2. In the **Value Type** list, select **Boolean**.
5. In the **Value Settings** section:
    1. In the **Type** list, select **String**, and enter `Secret` in the field.
6. In the **Timeout and Retry** section:
    1. In the **Request Timeout (ms)** field, enter `2000`.
7. Click **Save**.

Next steps

Create attributes for the `client_id`, `client_secret`, and active token.

Creating attributes for the `client_id`, `client_secret`, and active token

Create the attributes you'll reference in the access token validation policy.

Steps

1. In the PingAuthorize administrative console, go to **Trust Framework > Attributes**.
2. In the **PingFed Client** section, create a `client_id` attribute:
    1. In the **Parent** list, select **PingFed Client**.
    2. In the **Value Settings** section:
        1. Select the **Default Value** checkbox and enter the client ID from the OAuth resource server that you want to use to validate the token in the field.

            For example, **authroize_rs**.
        2. In the **Type** list, select **String**.
    3. In the **Caching** section:
        1. In the **Cache Strategy** list, select **No Caching**.
3. In the **PingFed Client** section, create a `client_secret` attribute.
    1. In the **Parent** list, select **PingFed Client**.
    2. In the **Value Settings** section:
        1. Select the **Default Value** checkbox and enter the corresponding client secret from the OAuth resource server that you want to use to validate the token in the field.

2. In the **Type** list, select **String**.

3. In the **Caching** section:

1. In the **Cache Strategy** list, select **No Caching**.

4. Go to the **SCIM2** section, and create a `TokenActive` attribute to resolve the `PingFederate Introspection` service:

1. In the **Resolvers** section:

1. In the **Resolver type** list, select **Service**, then select **PingFederate introspection**.

2. In the **Value Settings** section:

1. In the **Type** list, select **Boolean**.

3. In the **Caching** section:

1. In the **Cache Strategy** list, select **No Caching**.

Next steps

Create a policy to validate the access token.

Creating policies to validate and authorize the access token

Create policies to validate the access token and determine authorization requirements.

Steps

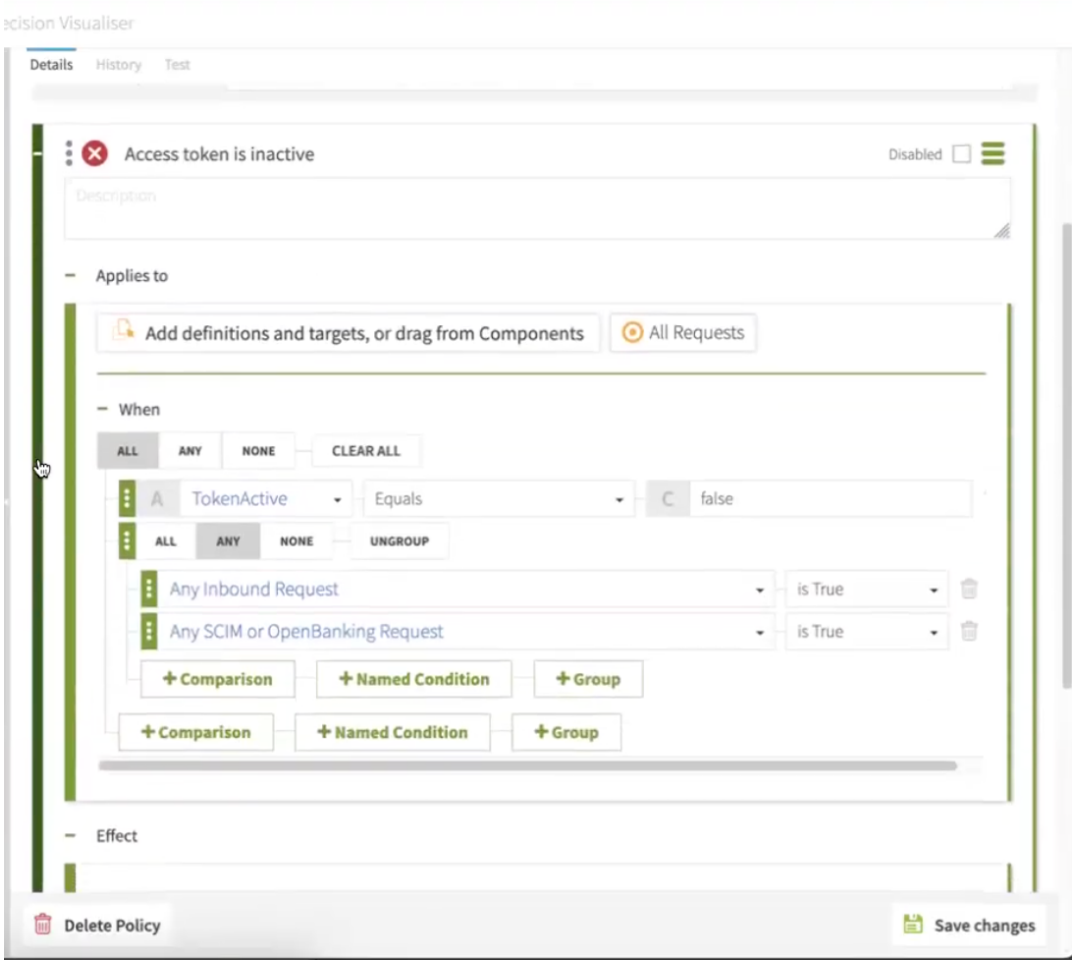
1. In the PingAuthorize Policy Editor, go to the **Policies** tab.
2. Go to the **Global Decision Endpoint** section and create a new policy called `Token Validation`.
3. In the **Rules** section, in the **Combining Algorithm** list, select **Unless one decision is deny, the decision will be permit**.
4. In the **Rules** section, add an **Access token is inactive** rule:

1. In the **Applies to** section, select **Add definitions and targets, or drag from components** and **All requests**.

2. In the **When** section, map the following:

1. Select **All**, then select **TokenActive**, **Equals**, and **False**.

2. Select **Any**, then set **Any Inbound Request** to **is True**, and **Any SCIM or OpenBanking Request** to **is True**.



5. In the **Statements** section:
    1. Add an **Invalid Token** statement:
        1. In the **Code** field, enter `denied-reason`.
        2. In the **Applies To** field, enter `Deny`.
        3. In the **Applies If** field, enter `All decisions in path match`.
        4. In the **Payload** field, enter `{"status":401, "message":"invalid_token","detail":"Access token is expired or otherwise invalid"}`.
        5. Make sure the **Obligatory** checkbox is selected.
6. Click **Save**.

Adding rate limiting rules

Add a rate limiting rule to prevent a client from overloading the server with too many requests in a specified period of time.

About this task

This rule uses a token bucket to control the number of incoming requests.

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, which is 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.

If a request wasn't received, the number of allowed requests is incremented by one at the end of each **Request Interval**. This continues until the number of allowed requests equals the **Max Burst Requests** value.

> **Note**
>
> Using the rate limiting rule in a clustered PingAccess environment can impose stricter clock synchronization requirements for requests processed by multiple engine nodes. Alternatively, you can configure a load balancer sitting in front of a PingAccess cluster to stick the session to a specific engine, ensuring that the rate limiting rule is applied by a single PingAccess engine node. For more information, see Configuring load balancing strategies.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. In the **Type** list, 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, such as a PingAccess cookie or an OAuth token. This is the default value.
IP	Restricts the number of requests based on the source Internet Protocol (IP) address. The IP address used to apply this policy comes from the HTTP request's IP source configuration options, or any options that override that configuration, if those options are configured.

Policy Granularity	Definition
OAuth Client	Restricts the number of requests to all OAuth tokens obtained by a specific **Client ID**. **Note** If you select OAuth Client as the policy granularity for a rate limiting rule, the rule will not work if it's applied to a web application.

6. Define, in milliseconds, a **Request Interval**.
7. **Optional:** Click **Show Advanced Settings** to configure advanced options:

Advanced Setting	Description
Max Burst Requests	If more than one request should be allowed in a request interval, 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 are allowed, regardless of this setting.
Set Retry-After Header	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** check box.

Advanced Setting	Description
Rejection Handling	To configure rejection handling, select a rejection handling method: ◦ If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL. ◦ If you select **Basic**, 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. If you select **Basic**, provide the following: 1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails. The default is `403`. 2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails. The default is `Forbidden`. 3. In the **Error Response Template File** field, enter 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. 4. In the **Error Response Content Type** list, select the type of content for the error response. This lets the client properly display the response.

8. Click **Save**.

Adding redirect rules

Add a redirect rule to specify a 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.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **Redirect**.
5. In the **Response Status Code** field, specify the response status code you want to associate with the redirect.

   The default is `302`.
6. In the **URL** field, specify the URL to which you want to redirect requests.

   You can use an absolute or a relative URL. If you use an absolute URL, it must include the `http/https` prefix. If you use a relative URL, it must begin with a slash. The URL can be specified with or without defining the port.
7. To opt out of automatic URL encoding, deselect the **Encode URL** check box.

   Learn more in PingAccess 8.1 (June 2024).
8. Click **Save**.

Adding rejection rules

Add a rejection rule to specify 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. You can either:

- Reject the request and display an error template.
- Redirect the user to another Uniform Resource Locator (URL) for error details, instructions, or additional actions.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **Rejection**.
5. In the **Rejection Handler** field, specify the rejection handler you want to use for the rule.
6. Click **Save**.

Adding time range rules

Add a time range rule, which examines a request and determines whether to grant access to a backend target site based on the request falling within a defined time frame.

About this task

You can use this rule when you want to restrict access to specific endpoints for certain time periods, such as during the work day from 8:00 a.m. to 5:00 p.m.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **Time Range**.
5. In the **Start Time** field, enter the beginning time for the time frame, such as 8:00 a.m.
6. In the **End Time** field, enter the ending time for the time frame, such as 5:00 p.m.

   > **Note**
   >
   > If you are using Internet Explorer or Firefox, you must enter the time in 24-hour format. For example, 5:00 p.m. is 17:00.
7. If access is not allowed when a match is found, select the **Negate** check box.
8. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

   Choose from:

   - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
   - If you select **Basic**, 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. If you select **Basic**, provide the following:

     1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

        The default is `403`.
     2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

        The default is `Forbidden`.

3. In the **Error Response Template File** field, enter 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.
4. From the **Error Response Content Type** list, select the type of content for the error response.

   This lets the client properly display the response.

9. Click **Save**.

Adding web session attribute rules

Use a web session attribute rule to examine a request and determine whether to grant access to a target site based on a match found between the attribute values in the PingAccess token and the attribute values that you specify with this rule.

About this task

Use this rule to test if an attribute in a web session contains a specific value, then grant or deny access to a target site based on whether a match is found.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Web Session Attribute**.
5. To grant the client access, select the **Attribute Name** that you want to match, such as `Group`.
6. In the **Matching Strategy** list, select the context that you want PingAccess to evaluate requests with when looking for a match.

   Choose from:

   - **Case-Sensitive**: To register as a match, the attribute value in the request must be written in the same case as the attribute value that you specify in step 7. By default, PingAccess uses this matching strategy.
   - **Case-Insensitive**: Case doesn't matter when looking for a match. Select this option for more flexibility if you might make changes to the attribute source that don't alter it semantically.
   - **DN Matching**: Normalizes both the attribute value that you specify in step 7 and any attribute value that PingAccess gathers at runtime from the user identity attributes as an X.500 distinguished name (DN). PingAccess then looks for a match between the distinguished names.

> **Tip**
>
> - If you select **DN Matching**, make sure to select an **Attribute Name** in step 5 that can contain a DN. The administrative console doesn't provide a warning if you select an invalid attribute type, but you can check your log files to confirm that you don't have any DN-related errors.
> - PingAccess validates the **Attribute Value** that you specify in step 7 to make sure that it's a valid X.500 DN that follows RFC-1779 or RFC-2253. Copy-paste the attribute value to ensure accuracy.
> - Relative DNs that have non-printable or non-UTF-8 string values, such as email and domain component (DC) relative DNs, are case sensitive. Otherwise, relative DN values are case-insensitive.
> - If you have a relative DN with multiple attribute-value pairs separated by a plus sign (**+**) delimiter, order doesn't matter because of normalization. For example, if you define part of a relative DN in the order `CN=Engineering+OU=Groups` , then a request containing those same attribute-value pairs in reverse order `OU=Groups+CN=Engineering` still returns a match.
> - The normalization process also replaces semicolon delimiters with commas, removes leading and trailing white spaces, and replaces hexadecimal values with their associated characters.

7. Enter the **Attribute Value** for the attribute name, such as `Sales` .

   > **Tip**
   >
   > Copy-paste the attribute value to ensure accuracy.

   If the attribute has multiple values at runtime, the attribute value you specify here must match one of those values.

   If you are using PingFederate as the token provider, PingAccess obtains token attributes from the PingFederate OpenID Connect (OIDC) policy attribute contract. Learn more in Configuring OpenID Connect Policies.

8. **Optional:** To add more attributes, click **Add Row**.
9. **Optional:** To remove a row, click the **Delete** icon.
10. **Optional:** To disallow access when a match is found, click **Negate**.

    > **Note**
    >
    > 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.

11. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

    Choose from:

    - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
    - If you select **Basic**, 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. If you select **Basic**, provide the following:

      1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

         The default is `403` .

2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

   The default is `Forbidden`.

3. In the **Error Response Template File** field, enter 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.

4. From the **Error Response Content Type** list, select the type of content for the error response.

   This lets the client properly display the response.

12. Click **Save**.

    To use this rule, select the **Request Profile** check box, indicating that you want PingAccess to request additional profile attributes from PingFederate when requesting the ID token.

Adding web session scope rules

Add web session scope rules, which examine the contents of the PingFederate validation response and determine whether to grant access to a backend target site based on a match found between the scopes of the validation response and the scope specified in the rule.

Before you begin

Support for the web session support rule might require the PingFederate access token to contain the scope `superuser`. To configure this, see Configuring access token attributes for superuser scope in PingFederate.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.

4. From the **Type** list, select **Web Session Scope**.
5. From the **Scope** list, select the scope you want to match to values returned from the access token.

   > **Note**
   >
   > This is one scope requirement in the set of scopes associated with the access token.

6. From the **Rejection Handler** list, select the rejection handler you want to associate with this rule.
7. Click **Save**.

Configuring access token attributes for superuser scope in PingFederate

A resource might require that the access token contains the scope `superuser`. Configure the `superuser` scope in PingFederate.

Steps

1. Enable Expressions within PingFederate.
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@encode(#this.get("context.OAuthScopes").getValuesAsHashSet(

4. Manage the OpenID Connect policy to add the following information:
    1. Attribute Contract— To extend the contract to include the `scope` attribute, select **Override Default Delivery** using the **ID Token**.

        > **Note**
        >
        > This step is not applicable to PingFederate 9.0 and earlier. Instead, in the **Manage Policy** window, select the **Include User Info in ID Token** check box.

    2. Attribute Scopes— From the **Scope** list, select `openid`, and from the **Attribute** list, select `scope`.

        > **Note**
        >
        > This feature does not exist in PingFederate versions earlier than 9.0. To work around this issue:
        > 1. Ensure PingAccess is configured to include `profile` in the list of **Web Session** scopes.
        > 2. In PingFederate, ensure the `profile` scope is defined in **Scope Management**.
        > 3. During authentication, the user must accept usage of the `profile` scope. If the user does not accept usage of the `profile` scope, then the web session scope rule will always fail for that user.

    3. Contract Fulfillment— Modify the `scope` **Attribute Contract** to use `Access Token` as the **Source** with a **Value** of `scope`.

Adding WebSocket handshake rules

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.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **WebSocket Handshake**.
5. In the **Allowed Origins**, enter one or more origins.

   If no origins are defined, all cross-origin WebSocket requests are denied.

   > **Important**
   >
   > Avoid using a value of `*` in this field. While this is a valid configuration, it is considered 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. To configure rejection handling, click **Show Advanced Settings**, then select a rejection handling method.

   Choose from:

   - If you select **Default**, use the **Rejection Handler** list to select an existing rejection handler that defines whether to display an error template or redirect to a URL.
   - If you select **Basic**, 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. If you select **Basic**, provide the following:

     1. In the **Error Response Code** field, enter the HTTP status response code to send if rule evaluation fails.

        The default is `403`.
     2. In the **Error Response Status Message** field, enter the HTTP status response message to send if rule evaluation fails.

        The default is `Forbidden`.
     3. In the **Error Response Template File** field, enter 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.
     4. From the **Error Response Content Type** list, select the type of content for the error response.

        This lets the client properly display the response.

9. Click **Save**.

Creating processing rules

Create processing rules to control how requests are processed in PingAccess.

For more information, see the following topics:

- Adding a cross-origin request rule
- Adding OAuth token cache time to live rules
- Adding PingAuthorize response filtering rules
- Rewrite rules overview

Adding a cross-origin request rule

Use cross-origin resource sharing (CORS) to let a web server grant access to restricted resources, such as fonts, JavaScript, and images, to an application that's served by another domain. This is done 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 client's request includes credentials. This pre-flight request is used to determine if the target server will permit the originating web server to process CORS requests.

PingAccess can evaluate the headers provided in a CORS request to grant or deny access to resources.

> **Tip**
>
> If the target application has an **Application Type** of `API`, you can allow the protected application to handle the request instead of PingAccess.
> 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. Make sure to set the **Methods** field to `OPTIONS` and clear the **Anonymous** option. This configuration allows the API request to be handled anonymously.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Cross-Origin Request**.

5. In the **Allowed Origins** field, enter one or more origin values.
    1. Click **+ New Value** to add additional values.

    > **Important**
    >
    > Avoid using a value of `*` in this field. While this is a valid configuration, it is an insecure practice.

6. (Optional) To configure additional options, click **Show Advanced Settings**.
    1. To permit user credentials to be used in determining access, enable **Allow Credentials**.
    2. If you entered a wildcard in the **Allowed Origins** field, select the **Mask Wildcard Policy** checkbox to replace the `Access-Control-Allow-Origin` response header with the value provided in the request's `Origin` header.
    3. 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 **Delete** icon.

        The default headers are `Authorization`, `Content-Type`, and `Accept`.
    4. To respond to CORS preflight requests with the expected response header, `Access-Control-Allow-Private-Network: true`, select **Allow Private Access Network**.

    > **Important**
    >
    > Google Chrome CORS preflight requests will soon include a new request header: `Access-Control-Request-Private-Network: true`. If preflight requests that contain this header do not receive a `Access-Control-Allow-Private-Network: true` header in response, access requests will be denied. Learn more about these headers in https://developer.chrome.com/blog/private-network-access-preflight?hl=en#what_to_do_if_your_website_is_affected.

    5. To make specific response headers available to the client that originated the cross-origin request, enter the headers in the **Exposed Response Headers** field.
    6. To add additional headers to the list, click **+ New Value**.
    7. To define the request methods allowed in cross-origin requests, enter the desired overrides in the **Overridden Request Methods** field.
    8. To modify the amount of time that the pre-flight `OPTIONS` request is cached, enter the maximum age (in seconds) in the **OPTIONS Cache Max Age** field.

        The default is `600` seconds.

7. Click **Save**.

Adding OAuth token cache time to live rules

Add an OAuth token cache time to live rules, which configures 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.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.
4. From the **Type** list, select **OAuth Token Cache Time to Live**.
5. If you want to cache the introspection of access tokens, click **Cache Tokens**.
6. In the **Time to Live (s)** field, 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 PingAuthorize response filtering rules

Add a response filtering rule, which contacts PingAuthorize or PingOne Authorize for filtering information.

> **Note**
>
> Limitations:
>
> - If you're using this rule with PingOne Authorize, detailed request context isn't available during response processing. As a result, response filtering can't be performed with the `PingOne.API Access Management.Identity.Access Token` attribute.

Before you begin

- Create a third-party service with PingAuthorize configured as the target. Learn more in Adding third-party services.
- Confirm that you are not using the agent model. PingAuthorize response filtering rules aren't available for agent deployments.

Important

Ensure that the sideband API setting `request-context-method` is set to `request` in PingAuthorize. Learn more about how to set this property and why it is necessary in Request context configuration in the PingAuthorize documentation.

About this task

A response filtering rule can modify the response given by PingAccess, based on the response from the PingAuthorize response application programming interface (API).

Important

The PingAuthorize sideband API cannot accept gzipped data from upstream server responses. To prevent the upstream server from sending compressed responses, ensure that upstream server requests add or replace the `Accept-Encoding` header with `Accept-Encoding: identity`.

To add a PingAuthorize response filtering rule:

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name, up to 64 characters long.

   Special characters and spaces are allowed.
4. In the **Type** list, select **PingAuthorize Response Filtering**.
5. In the **Third Party Service** list, select your PingAuthorize service.
6. In the **Shared Secret** field, enter the shared secret from PingAuthorize.
7. To include the HTTP request body in the HTTP request data sent to PingAuthorize, select the **Include Request Body** checkbox.

   If PingAuthorize needs the request body for an access decision, make sure that this checkbox is selected. Otherwise, clearing the checkbox could improve performance.

   This option is selected by default.
8. To include the HTTP response body in the HTTP response data sent to PingAccess, select the **Include Response Body** checkbox.

   If PingAuthorize needs the response body to modify the response that it gives to a user, make sure that this checkbox is selected. Otherwise, clearing the checkbox could improve performance.

   This option is selected by default.
9. (Optional) To configure advanced options, click **Show Advanced**:
   1. (Optional) In the **Sideband Endpoint** field, enter the sideband API endpoint location.
   2. (Optional) In the **Shared secret header name** field, enter a header in which to send the shared secret.

3. (Optional) In the **Additional Request Headers** section, enter a **Header Name** and **Header Value** for any additional headers that you want to include in the request to PingAuthorize. Click **+ Add Row** to add other headers as necessary.

   PingAuthorize can use the additional headers to determine the policy set that's most relevant to the request context.

   If an additional header that you configured appears in a user request, PingAccess replaces the original request header and its corresponding values with the **Header Value** that you configured. If you leave the **Header Value** field blank, PingAccess removes this header from the request to PingAuthorize.

   If the **Header Value** contains the substrings `"${APPLICATION_NAME}"` or `"${RESOURCE_NAME}"`, PingAccess replaces those strings with the name of the requested application or resource as defined in PingAccess.

4. (Optional) In the **Additional Response Headers** section, enter a **Header Name** and **Header Value** for any additional headers that you want to include in the modified response. Click **+ Add Row** to add other headers as necessary.

   PingAuthorize can use the additional headers to determine the policy set that's most relevant to the response context.

   If an additional header that you configured appears in the response, PingAccess replaces the original response header and its corresponding values with the **Header Value** that you configured. If you leave the **Header Value** field blank, PingAccess removes this header from the response.

   If the **Header Value** contains the substrings `"${APPLICATION_NAME}"` or `"${RESOURCE_NAME}"`, PingAccess replaces those strings with the name of the requested application or resource as defined in PingAccess.

10. Click **Save**.

Rewrite rules overview

PingAccess allows for the manipulation of the Request URI, the cookie domain, the cookie path, three of the response headers (`Location`, `Content-Location`, and `URI`), and the response content.

For example, a site is hosted on `https://server1.internalsite.com` under `/content/`. Users access the site through the following Uniform Resource Locator (URL) in their browser: https://server1.internalsite.com/content/

For demonstration purposes, assume 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 must rewrite the content. The information below discusses an example scenario.

> **Note**
>
> This conceptual overview 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 points to `https://server1.internalsite.com/content/` now points 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 backend 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/`.

  > **Note**
  >
  > 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 Uniform Resource Identifier (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 rewrite content rules

Add rewrite content rules, which modify text in HTTP response bodies as it is served to the client in PingAccess.

Before you begin

Confirm that you are not using the agent model. Rewrite content rules aren't available for agent deployments.

About this task

A rewrite content rule uses a subset of the Java regular expression syntax that excludes look-behind constructs, such as `\b`, and the boundary matcher, `\G`. If no Java regular expression syntax is used, the rule performs 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.

> **Note**
>
> Extensive use of rewrite content rules might 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 add a rewrite content rule:

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters long.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Rewrite Content**.
5. To define what type of response data to which the rewrite rule applies, enter one or more **Response Content-Types**.

   The default values are `text/html`, `text/plain`, and `application/json`. The list is an ordered list.

   > **Note**
   >
   > Only text-based content types are supported. Text-based content types compressed with gzip, deflate, or compress will decompress prior to rewrite rule processing, however, the content is not re-compressed before sending to the client.
6. Define one or more sets 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 unintentionally modifying content.

7. If the protected application does not return a **Content-Type** header, select **Missing Content-Type Allowed**.
8. If you enable **Missing Content-Type Allowed**, specify the encoding the application returns in the **Missing Content-Type Charset** field.

   This field could contain `UTF-8`. A list of valid values is available in the Oracle Java 8 SE Technical Note.
9. 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 you can specify is 1024 bytes. There is no maximum value.
10. Click **Save**.

Rewrite content rule examples

This table provides examples of rewrite content rule use cases and their results.

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
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 rewrite cookie domain rules

Add a rewrite cookie domain rule, which converts the cookie domain returned by the site into a public-facing domain, in PingAccess.

Before you begin

Confirm that you are not using the agent model. Rewrite cookie domain rules aren't available for agent deployments.

About this task

When a site places a cookie on a cookie domain such as `internalsite.com`, or `.internalsite.com`, PingAccess rewrites the `Domain` portion of the `Set-Cookie` response header with a public-facing domain such as `publicsite.com`, or `.publicsite.com`, using the information configured in the rewrite cookie domain rule.

> **Note**
>
> 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 add a rewrite cookie domain rule:

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Rewrite Cookie Domain**.
5. If you need to explicitly define the target host, clear the **Any Site Target Host** check box.

When you select the **Any Site Target Host** check box, PingAccess will rewrite the cookie domain if it is set to the domain defined in a site's target host list.

6. If you clear the **Any Site Target Host** check box, enter the domain name used by the backend site in the **Server-Facing Cookie Domain** field.
7. In the **Public-Facing Cookie Domain** field, enter the domain name that you want to display in the response from PingAccess.
8. Click **Save**.

Adding rewrite cookie path rules

Add a rewrite cookie path rule, which converts the cookie path returned by the site into a public-facing path, in PingAccess.

Before you begin

Confirm that you are not using the agent model. Rewrite cookie path rules aren't available for agent deployments.

About this task

This task 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 add a rewrite cookie path rule:

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Rewrite Cookie Path**.
5. In the **Server-Facing Cookie Path** field, enter the path name where the cookie is valid for the backend site.
6. In the **Public-Facing Cookie Path** field, enter the path name that you want to display in the response from PingAccess.
7. Click **Save**.

Adding rewrite response header rules

Add a rewrite response header rule, which converts the response header value returned by the site into a public-facing value.

Before you begin

Confirm that you are not using the agent model. Rewrite response header rules aren't available for agent deployments.

About this task

This rule rewrites one of three response headers:

1. `Location`
2. `Content-Location`
3. `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 add a rewrite response header rule:

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name up to 64 characters.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Rewrite Response Header**.
5. If the target host needs to be explicitly defined, clear the **Any Site Target Host** check box.

   When you select the **Any Site Target Host** check box, PingAccess will rewrite the response header URI if it contains a domain defined in a site's target host list.
6. If you clear the **Any Site Target Host** check box, enter the domain name used by the backend site in the **Server-Facing URI** field.
7. In the **Public Path** field, enter a valid URI path that you want to write into the URI.

   This must be a valid Uniform Resource Identifier (URI) path and begin and end with a forward slash (`/`).

   Example:

   `/importantstuff/` or `/`
8. Click **Save**.

Adding rewrite URL rules

Add a rewrite URL rule, which examines the URL of every request and determines if a pattern matches, in PingAccess.

Before you begin

Confirm that you are not using the agent model. Rewrite URL rules aren't available for agent deployments.

About this task

When you define a regular expression in a rule (such as `regex`), and if a pattern matches, PingAccess uses the information configured in the rewrite URL rule and rewrites that portion of the Uniform Resource Locator (URL) into a path that the site can understand.

To add a rewrite URL rule:

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name for the rule up to 64 characters.

   Special characters and spaces are allowed.
4. In the **Type** list, select **Rewrite URL**.
5. In the **Map From** field, enter the `regex` of the URL path and the query you want to match.

   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** field, enter the URL path and the query that you want to generate.

   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**.

Rewrite URL rule configuration examples

This table displays four examples of rewrite URL rule configurations in PingAccess.

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`

Map from value	Map to value	Example request	Rewrite by PingAccess
`/bank/index.html`	`/application/index.jsp`	`/bank/index.html?query=stuff`	`/application/index.jsp?query=stuff`

Editing rules

Edit the properties of an existing rule in PingAccess.

Steps

1. Click **Access**, then go to **Rules** > **Rules**.
2. Click to expand the rule you want to edit.
3. Click the **Pencil** icon.
4. Make the desired changes to the rule you want to edit.
5. To confirm your changes, click **Save**.

Deleting rules

Delete an existing rule set in PingAccess.

Steps

1. Click **Access**, then go to **Rules** > **Rules**.
2. Click to expand the rule you want to delete.
3. Click the **Delete** icon.
4. To confirm your change and delete the rule, click **Delete**.

Rule sets

Rule sets let you combine rules into reusable groupings in PingAccess.

For more information on what you can do with rule sets, see:

- Adding rule sets
- Editing rule sets
- Deleting rule set groups

Adding rule sets

Add a new rule set in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rule Sets**.
2. Click **+ Add Rule Set**.
3. In the **Name** field, enter a name for the rule set.

   Special characters and spaces are allowed.
4. In the **Success Criteria** section, select one of the following options:

   All

   Requires all rules in the set to succeed.

   > **Note**
   >
   > When **Success Criteria** is set to **All**, the first rule in the set that fails establishes the error handling.

   Any

   Requires only one of the rules in the set to succeed.

   > **Important**
   >
   > If there are any processing rules in the rule set, PingAccess flags those rules with a tooltip warning that PingAccess won't process additional rules if the first rule in the set succeeds.
   > This is because when **Success Criteria** is set to **Any** for a rule set, the first processing rule that succeeds causes PingAccess not to evaluate all other rules in the set.
   > If you want to use processing rules on protected applications and handle access control decisions using **Any** criteria, assign processing rules directly to the application or create a separate rule set for the processing rule sets that uses **All** criteria.
5. Add one or more rules to the rule set you're creating:
   1. In the `Available Rules` section, click the **+** icon to add the desired group to the **Selected Rules** section.

      > **Tip**
      >
      > - To create a new rule, click **+ Create Rule**. You can find more information in Rules.
      > - Use the **Search Available Rules** search bar to expedite finding a specific rule.
      > - You can click and hold to drag a rule directly from the **Available Rules** section to the **Selected Rules** section. This expedites configuring the desired hierarchy of the rule set you're creating.
6. In the **Selected Rules** section, configure the hierarchy by clicking and holding to drag a specific rule to the desired position in the configuration.

   > **Tip**
   >
   > - Processing occurs from top to bottom.
   > - To remove a rule from the configuration, click the **−** icon.
7. To save the rule set, click **Save**.

Editing rule sets

Edit an existing rule set in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rule Sets**.
2. Click to expand the rule set you want to edit.
3. Click the **Pencil** icon.
4. To make the desired edits:

   Choose from:

   - Edit the rule set **Name** or **Success Criteria**.
   - To remove a rule from a rule set, click **-** next to the rule.
   - To re-order the rules, drag a rule within a rule set up or down.
5. To confirm your edits, click **Save**.

Deleting rule sets

Delete an existing rule set in PingAccess. You must remove all associations between the rule set and any applications or resources before you can delete it.

Steps

1. Click **Access**, then go to **Rules > Rule Sets**.
2. Click to expand the rule set you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Rule set groups

Rule set groups let you combine one or more rule sets into reusable groups in PingAccess.

For more information on what you can do with rule set groups, see:

- Adding rule set groups
- Editing rule set groups
- Deleting rule set groups

Adding rule set groups

Add a new rule set group in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rule Set Groups**.
2. Click **+ Add Rule Set Group**.
3. In the **Name** field, enter a name for the rule set group.

   Special characters and spaces are allowed.
4. In the **Success Criteria** section, select one of the following options:

   All

   Requires all rule sets in the group to succeed.

   > **Note**
   >
   > When **Success Criteria** is set to **All**, the first rule set in the group that fails establishes the error handling.

   Any

   Requires only one of the rule sets in the group to succeed.

   > **Note**
   >
   > When **Success Criteria** is set to **Any**, the last rule set in the group establishes the error handling.
5. To add one or more rule sets to the group you're creating:
   1. In the `Add From` section, select **Rule Sets**.
   2. In the `Available Rule Sets` section, click the **+** icon to add the desired rule set to the **Selected Rule Sets and Groups** section.

      > **Tip**
      >
      > - To create a new rule set, click **+ Create Rule Set**. You can find more information in Adding rule sets.
      > - To expedite finding a specific rule set, use the **Search Available Rule Sets** search bar.
      > - You can click and hold to drag a rule set directly from the **Available Rule Sets** section to the **Selected Rule Sets and Groups** section. This expedites configuring the desired hierarchy of the rule set group you're creating.
6. To add one or more rule set groups to the group you're creating:
   1. In the `Add From` section, select **Rule Set Groups**.
   2. In the `Available Rule Set Groups` section, click the **+** icon to add the desired group to the **Selected Rule Sets and Groups** section.

> **Tip**
>
> - Use the **Search Available Rule Set Groups** search bar to expedite finding a specific group.
> - You can click and hold to drag a group directly from the **Available Rule Sets Groups** section to the **Selected Rule Sets and Groups** section. This expedites configuring the desired hierarchy of the rule set group you're creating.

7. In the **Selected Rule Sets and Groups** section, configure the hierarchy by clicking and holding to drag a specific rule set or rule set group to the desired position in the configuration.

> **Tip**
>
> - Processing occurs from top to bottom.
> - To remove a rule set or rule set group from the configuration, click the **—** icon.

8. To save the rule set group, click **Save**.

Editing rule set groups

Edit the properties of an existing rule set group in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rule Set Groups**.
2. Click to expand the rule set group you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the rule set groups:
    1. Edit the rule set group **Name** or **Success Criteria**.
    2. To add a new rule set or rule set group to the existing rule set group, drag them into the selected column.
    3. To create a new rule set, click **+ Create Rule Set**

       The new rule set is not automatically added to the rule set group.
    4. To reorder the rules, drag a rule set within a rule set group up or down.
    5. To remove a rule set from a rule set group, click **-** to the right of any rule set.
5. To confirm your changes, click **Save**.

Deleting rule set groups

Delete an existing rule set group in PingAccess. You must remove any associations between the rule set group and any applications or resources before you can delete it.

Steps

1. Click **Access**, then go to **Rules > Rule Set Groups**.

2. Click to expand the rule set group you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

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 to send if policy evaluation fails.

You include a rejection handler in a rule by clicking to expand **Advanced** on the **Create Rule** window.

PingAccess contains three predefined rejection handlers:

Default API Rejection Handler

Returns a `403` status code in a JavaScript Object Notation (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 an HTML template.

Creating rejection handlers

Create a new rejection handler in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rejection Handlers**.
2. Click **+ Add Rejection Handler**.
3. In the **Name** field, enter a name for the object.
4. Choose the **Type**:

   Choose from:

   - **Error Template**
   - **Redirect**

   1. If you selected **Error Template**, specify the **Response Status Code**, the **Template File**, and the **Content Type**.
   2. If you selected **Redirect**, specify the **Response Status Code** and **URL** to which you want to redirect if policy evaluation fails.

You can use an absolute or a relative Uniform Resource Locator (URL). If you use an absolute URL, it must include the `http/https` prefix. If you use a relative URL, it must begin with a slash. The URL can be specified with or without defining the port.

> **Tip**
>
> To opt out of automatic URL encoding, deselect the **Encode URL** checkbox. Learn more in PingAccess 8.1 (June 2024).

Editing rejection handlers

Edit the properties of an existing rejection handler in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rejection Handlers**.
2. Click to expand the rejection handler you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the rejection handler.
5. To confirm your changes, click **Save**.

Deleting rejection handlers

Delete an existing rejection handler in PingAccess.

Steps

1. Click **Access**, then go to **Rules > Rejection Handlers**.
2. Click to expand the rejection handler you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Authentication

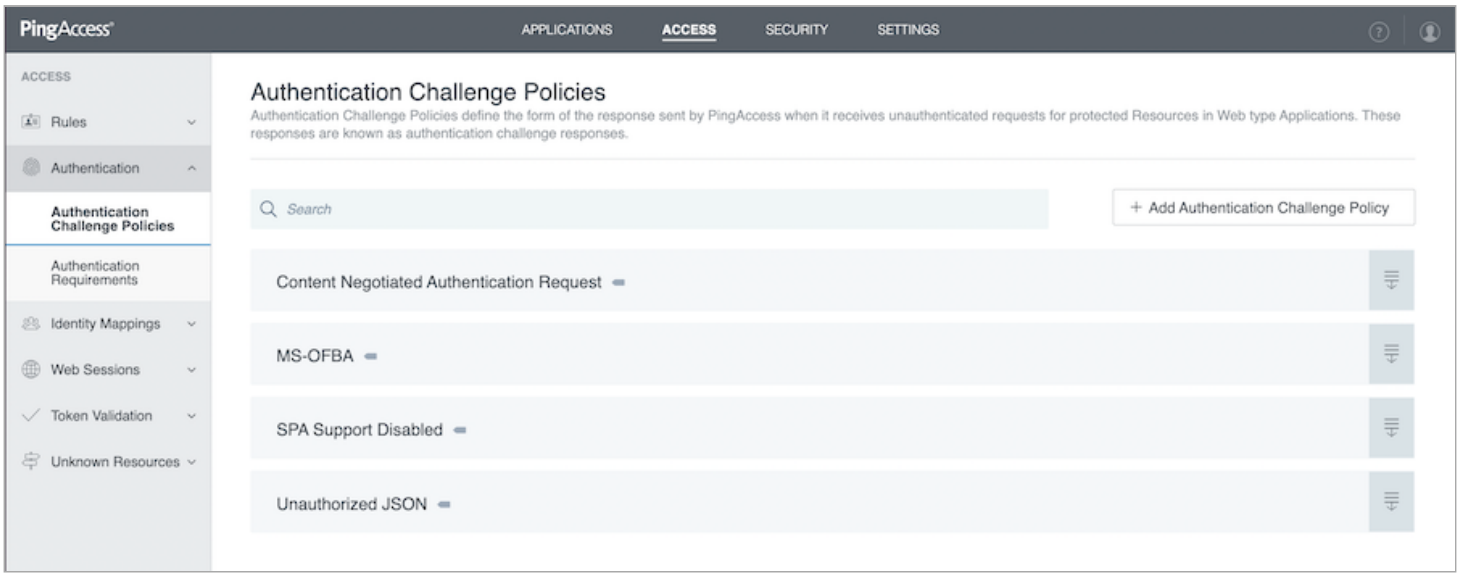
Authentication challenge policies and authentication requirements let you control how users are authenticated.

Authentication challenge policies

Authentication challenge policies set the response sent by PingAccess when it receives unauthenticated requests for protected resources from web applications.

Each authentication challenge policy consists of one default mapping and zero or more challenge response mappings. When a user attempts to access a protected resource and a PingAccess web session has not yet been established, and when the request characteristics match those of a challenge response mapping, the response specified in the challenge response mapping is used. If the request does not match any challenge response mappings, the default mapping is used.

> **Note**
>
> PingAccess deploys several system-provided authentication challenge policies that are automatically enabled on initial startup. These policies are identifiable by the presence of a gray flag to the right of the policy's name on the **Authentication Challenge Policies** page. You can view system-provided authentication challenge policies, but you can't edit or delete them. To create your own customizable authentication challenge policies, see Configuring authentication challenge policies.



System-Provided Authentication Challenge Policies

System-Provided Authentication Challenge Policy	Description
Content Negotiated Authentication Request	Allows the user agent to negotiate the form of the authentication challenge response with an Accept header field in the request. If the user agent requests HTML, a 401 response returns with an HTML body that will automatically initiate an OpenID Connect (OIDC) login flow via a JavaScript redirect. Otherwise, the user agent receives a JavaScript Object Notation (JSON) response.

MS-OFBA	Provides MS-OFBA support, enabling you to open Microsoft (MS) Office documents protected by PingAccess in an in-app browser that redirects to the OpenID Provider (OP) for user authentication. If the user authenticates successfully, PingAccess establishes a web session and redirects the user to the MS Office application that matches the document type (spreadsheets open in Microsoft Excel, for example). **Note** The system-provided **MS-OFBA** authentication challenge policy does not work with any Microsoft Office applications running on macOS. The macOS in-app browser is much more restrictive than the one in Windows. It cannot set the nonce cookie that PingAccess requires before redirecting a user to the OP. **Note** In some environments, Internet Explorer configurations can dictate the behavior of the in-app browser in Microsoft Office products. If the document you requested fails to download, ensure that **Do not save encrypted pages to disk** is disabled in **Internet Explorer → Internet Options → Advanced → Settings → Security**. Web sessions are not shared between different MS Office apps, but users do not have to reauthenticate for apps they've already opened. So, if you authenticate after opening an Excel sheet, you can open other Excel sheets without needing to reauthenticate. If you open a Word document after authenticating into an Excel sheet, however, then you must reauthenticate to access the Word document. **Note** You can configure MS-OFBA challenge response mappings and challenge response generators individually in a custom authentication challenge policy, but a custom creation like this is best used to address unusual circumstances as they come up. For example, if Microsoft includes a new entry on the list of user agents approved for MS-OFBA, an administrator can create a branching challenge response mapping for the new user agent and set it to trigger the **MS-OFBA Authentication Request Redirect** response generator. Alternately, an administrator could set the MS-OFBA header `X-FORMS_BASED_AUTH_DIALOG_SIZE` using the **Append Header Fields** challenge response filter.

System-Provided Authentication Challenge Policy	Description
SPA Support Disabled	PingAccess disables SPA support on a global scale out of the box so admins do not have to manually turn off SPA support on each individual application. SPA support can be re-enabled on a per-application basis, as seen in Application field descriptions. For more information on the benefits of SPA support, see Applications operations.
Unauthorized JSON	Unconditionally returns a 401 JSON response.

Authentication requirements

Authentication requirements are methods of authentication that are ordered by preference.

When a user attempts to access a PingAccess web application configured with an authentication requirement list containing the values password and certificate, 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 matches the requirements and redirects the user to the originally requested application with the PingAccess cookie set. When the user attempts to access a more sensitive application configured with an authentication requirement list containing the value (certificate), they are redirected to PingFederate to authenticate with a certificate.

You can configure applications with authentication requirement lists that have no overlap. For example, if one list has a password and another list has a certificate, a user navigating between applications might be required to authenticate each time they visit an application. When configuring authentication requirement lists to protect higher value applications with step-up authentication, include stronger forms of authentication when configuring lower value applications.

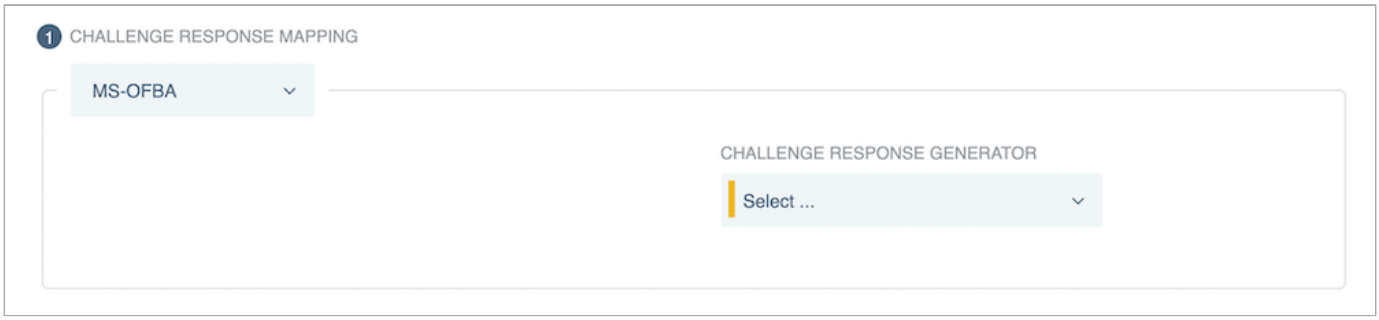
Configuring authentication challenge policies

Configure an authentication challenge policy in PingAccess to set the response that PingAccess sends when it receives unauthenticated requests for protected resources.

Steps

1. Click **Access**, then go to **Authentication > Authentication Challenge Policies**.
2. Click **+ Add Authentication Challenge Policy**.
3. In the **Name** field, enter a unique name for the authentication challenge policy.
4. **Optional:** In the **Description** field, enter a description for the authentication challenge policy.
5. In the **Challenge Response Mapping** list, select a mapping type:

   The **MS-OFBA** challenge response mapping examines `OPTIONS HTTP` method requests to determine if the user agent is a client that supports Microsoft's MS-OFBA protocol or if the request has a Boolean flag indicating that it supports MS-OFBA.

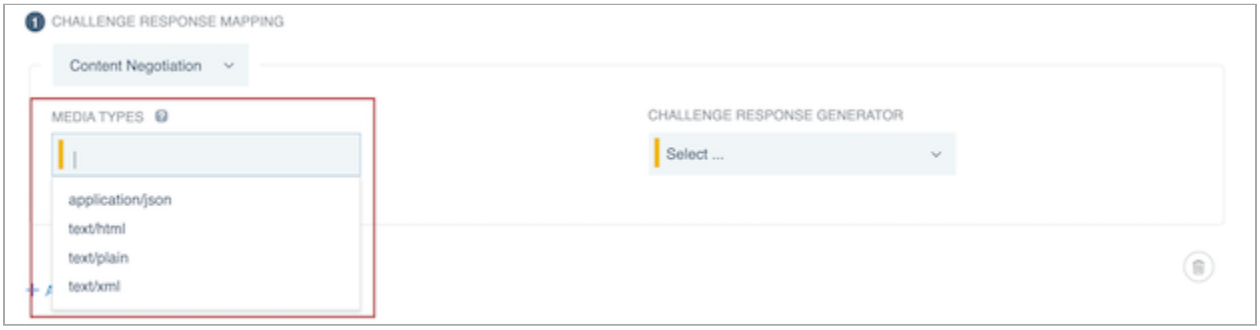


Tip

PingAccess provides an MS-OFBA authentication challenge policy that's included with the system by default. As such, the **MS-OFBA** challenge response mapping is best used to address edge cases as they come up. For more information, see Authentication.

1. If you selected the **Content Negotiation** mapping type, in the **Media Types** list, select one or more media types.

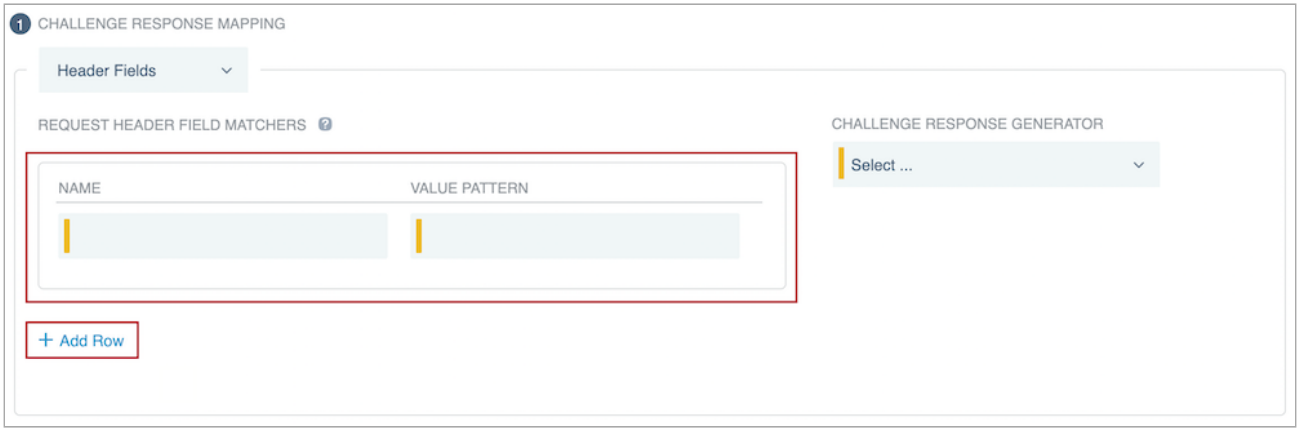
   The selection options for media types are **application/json**, **text/html**, **text/plain**, or **text/xml**.



   Result:

   If the `Accept` header field in the request matches any of the specified media types, the mapping is applied.

2. If you selected the **Header Fields** mapping type, click **+ Add Row** to add one or more rows, and then in the **Name** and **Value Pattern** fields, enter a name and value pattern for each row.



   Result:

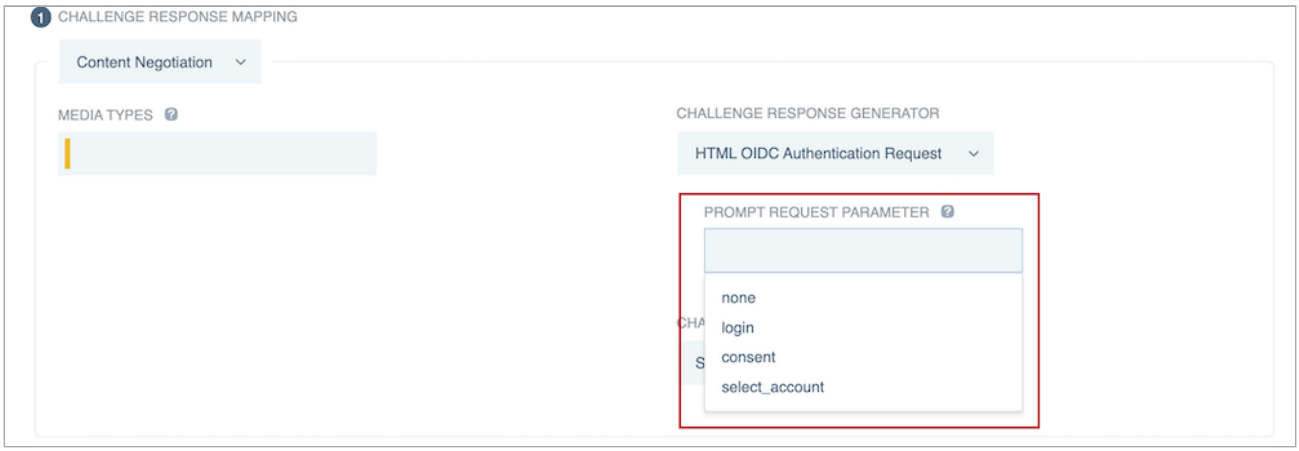
   If all of the specified header fields in the request match the specified value patterns, the mapping is applied.

6. Configure a challenge response generator for the challenge response mapping:

    1. In the **Challenge Response Generator** list, select a challenge response generator.

        For more information, see Authentication challenge responses and Authentication challenge response generator descriptions.

    2. **Optional:** If you selected **Browser-handled OIDC Authentication Request**, **HTML OIDC Authentication Request**, **MS-OFBA Authentication Request Redirect**, **OIDC Authentication Request Redirect**, or **PingFederate Authentication API Challenge**, you can select one of the following options from the **Prompt Request Parameter** list to let the authorization server know whether to prompt an end-user to reauthenticate or provide consent.



none

Returns an error if the end-user isn't authenticated or if the OAuth client doesn't have user consent for the requested claims. The authorization server doesn't prompt the end-user with a consent or authentication page if this option is selected.

login

The authorization server prompts the end-user to reauthenticate. If the end-user doesn't reauthenticate successfully, it returns an error.

> **Note**
>
> For extra security, PingAccess validates the <*auth_time*> the login request was sent at against the <*auth_time*> the OpenID Provider (OP) sets in the response.

consent

The authorization server prompts the end-user for consent before giving information to the OAuth client. If the end-user doesn't give their consent, it returns an error.

select_account

The authorization server prompts the end-user to specify which account they are using, in case they have multiple accounts. If the end-user doesn't select an account, it returns an error.

If you are using PingFederate as the OP, you should also enable push authorization requests on the web session you want to use with this authentication challenge policy. This advanced setting provides an additional layer of security against frontchannel tampering. For more information, see **Enable Push Authorization** in Creating web sessions.

> **Note**
>
> You can set the **Prompt Request Parameter** in two places: on the web session or on one of the OpenID Connect (OIDC) authentication challenge response generators. A value set on a specific authentication challenge response generator takes precedence over one set on a web session.

3. **Optional:** If you selected **OIDC Authentication Request Redirect**, **Redirect Challenge**, or **Templated Challenge**, you can configure PingAccess to let the authentication authority know why a user was redirected to it:

    1. Go to the **Web Sessions** page and expand the web session that you want to edit.

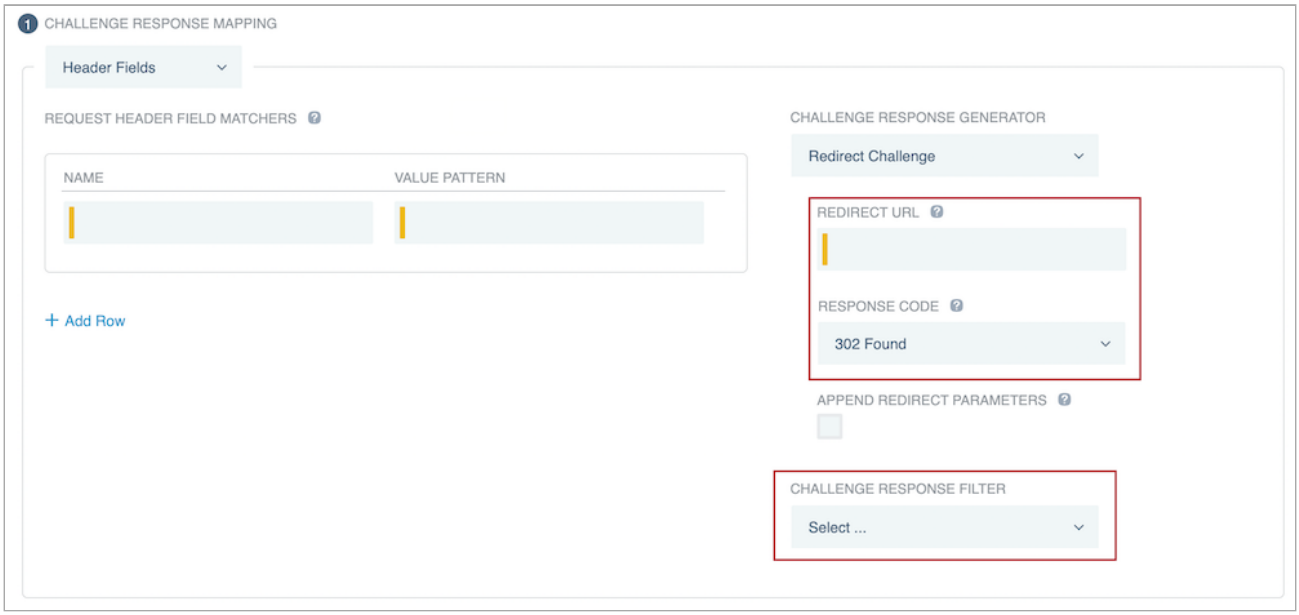
    > **Important**
    >
    > If you selected **Redirect Challenge**, make sure that you select the **Append Redirect Parameters** check box in step 6d.
    > If you selected **Templated Challenge** and are using PingFederate as an authentication source, make sure that you connect to PingFederate's redirectless OIDC flow.

    2. Click the **Pencil** icon, and then select the **Provide Authentication Feedback** check box under **Advanced Settings**.

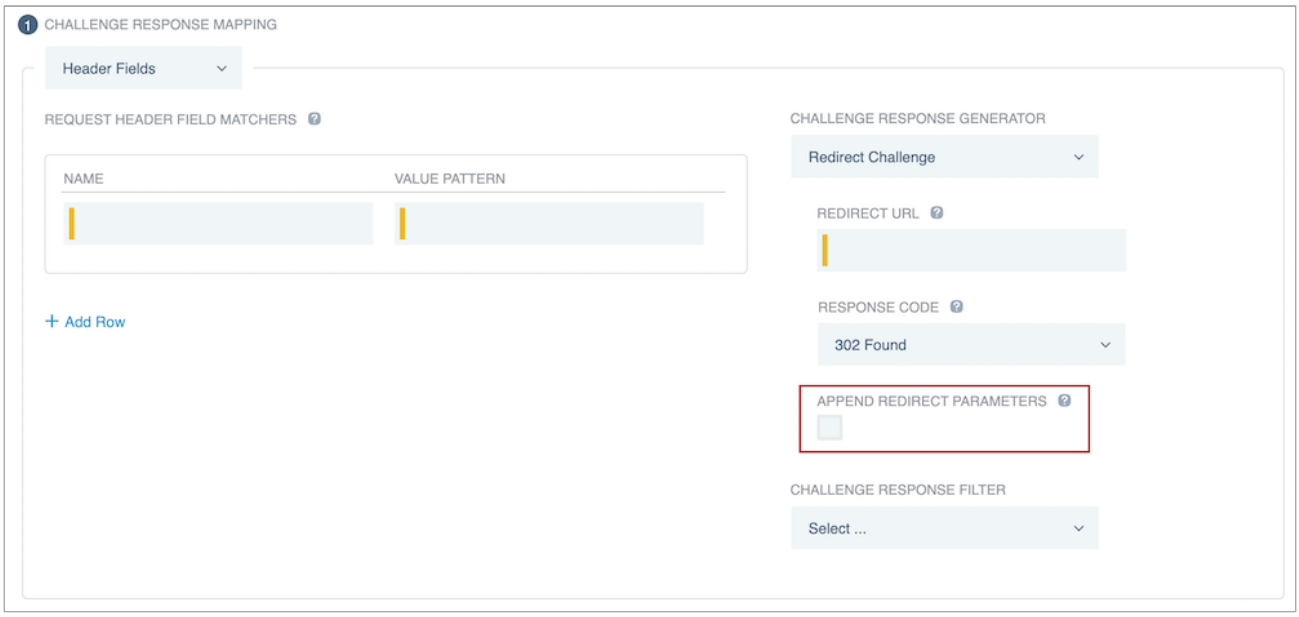
    For more information about the feedback PingAccess can provide, see Creating web sessions.

    An **OIDC Authentication Request Redirect** issues the feedback key `vnd_pi_authn_feedback`. The feedback key issued by an **Redirect Challenge** is `authnFeedback`, and the feedback key issued by an **Templated Challenge** response generator is `oidc.authnFeedback`.

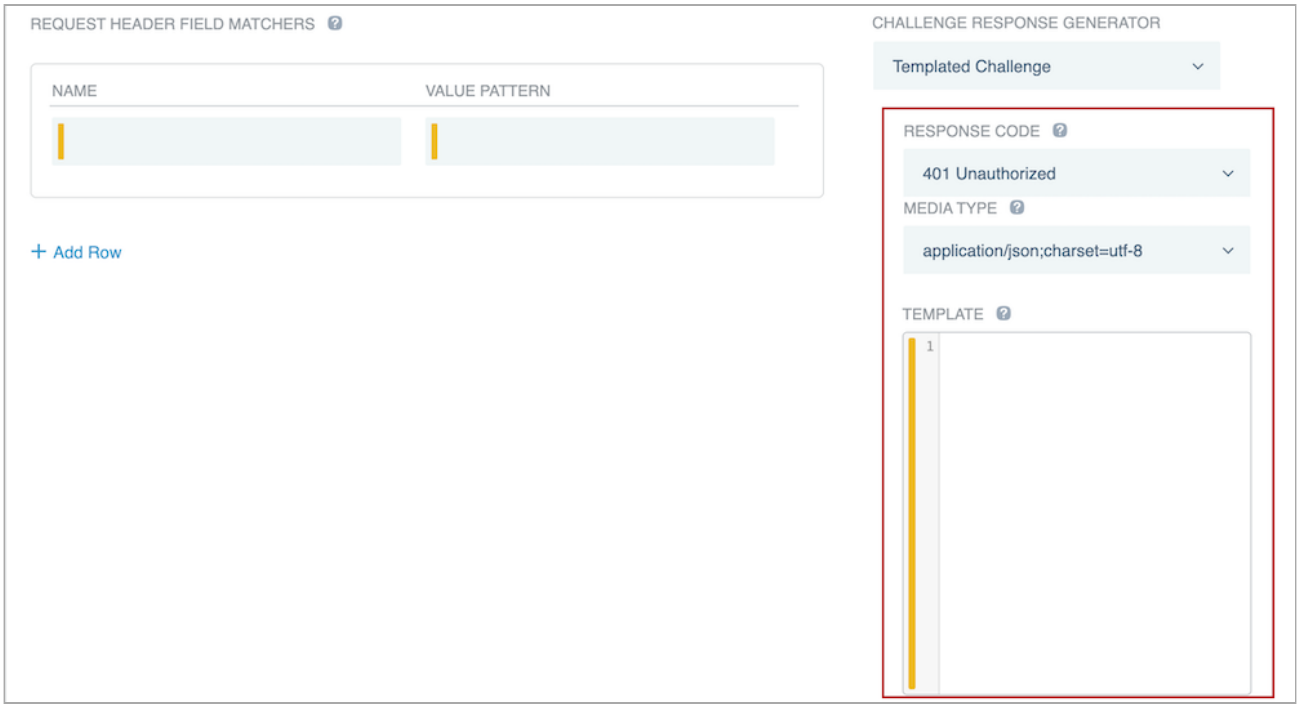
4. If you selected **Redirect Challenge**, enter a **Redirect URL** and select a **Response Code** for the redirect.



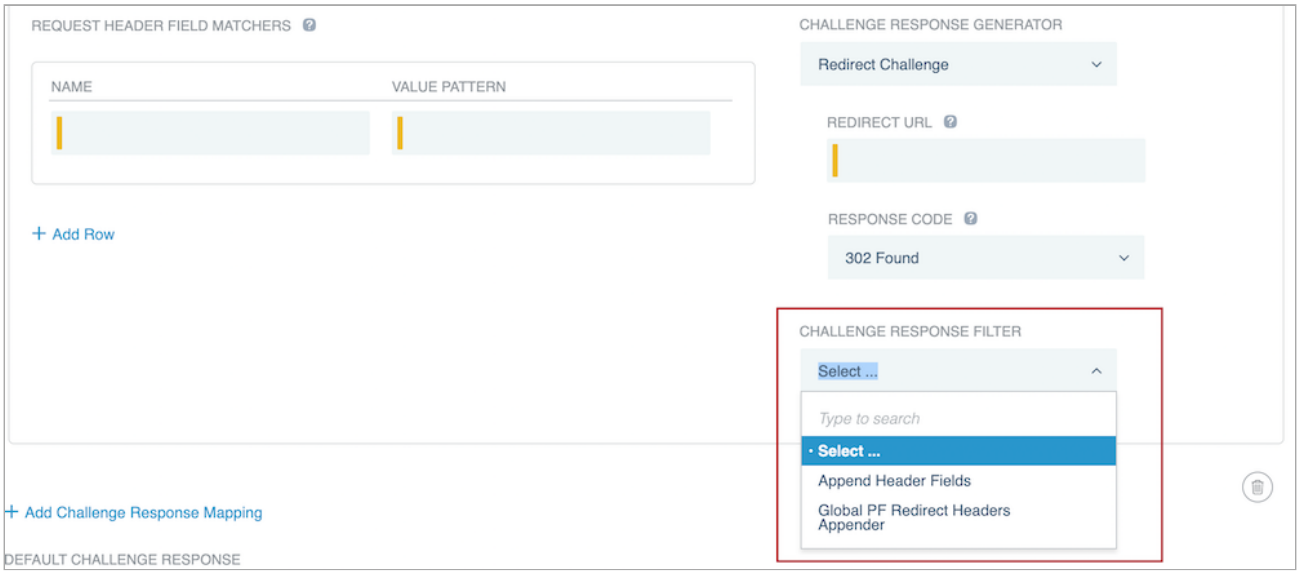
Optionally, select the **Append Redirect Parameters** check box to append PingFederate OIDC parameters and the Uniform Resource Locator (URL) of a requested resource within the query string of a redirect URL that you specify. For more information, see the **Redirect Challenge** table entry in Authentication challenge response generator descriptions.



5. If you selected **Templated Challenge**, select a **Response Code** and **Media Type** for the template, and then enter the template in the **Template** field.



6. In the **Challenge Response Filter** list, select a challenge response filter.



- If you selected **Append Header Fields**, click **+ Add Row**.

  Enter a **Name** and **Value** in each row.

- If you selected Global PF Redirect Headers Appender, then PingAccess will add the headers defined by the `pf.redirect.headers` in `<PA_HOME>/conf/run.properties` as described by the Configuration file reference.

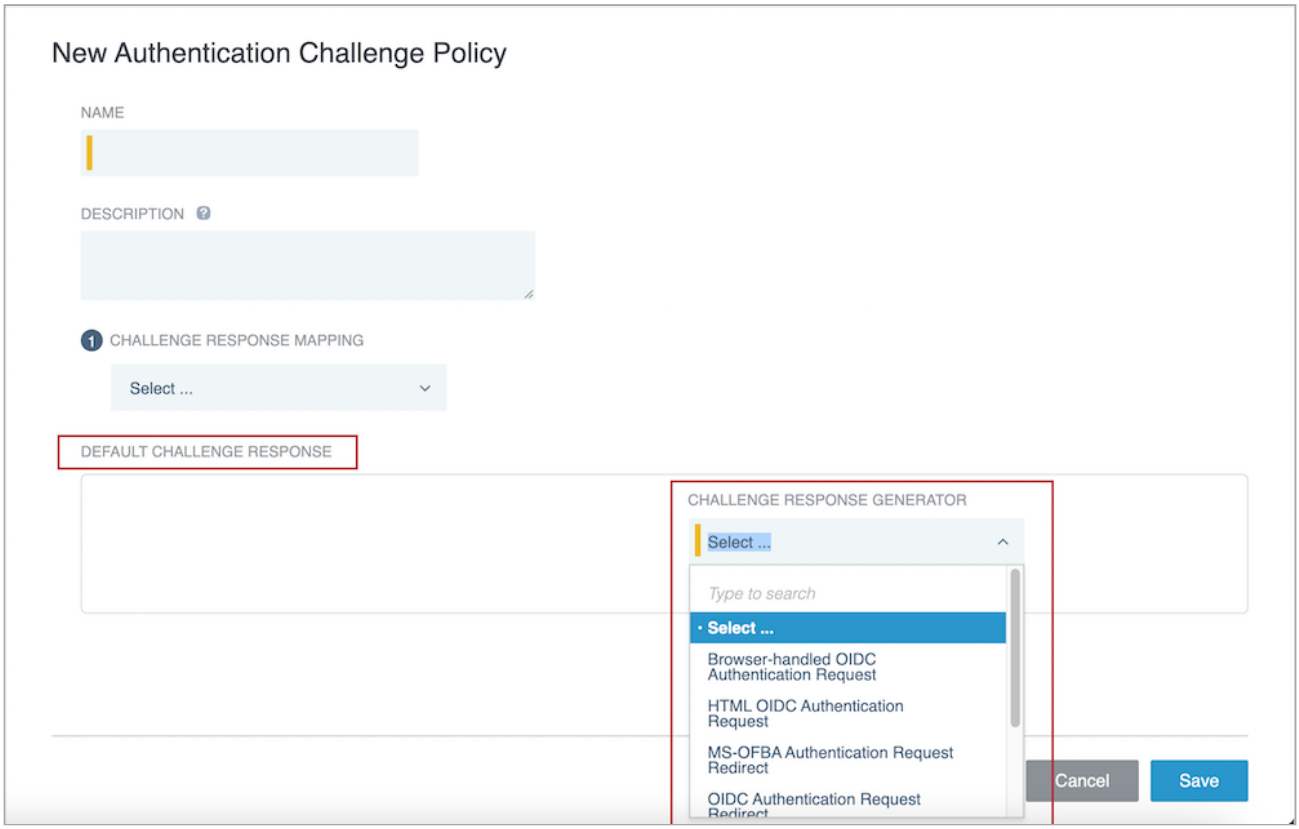
Result:

The specified HTTP response header fields are appended to the authentication challenge response.

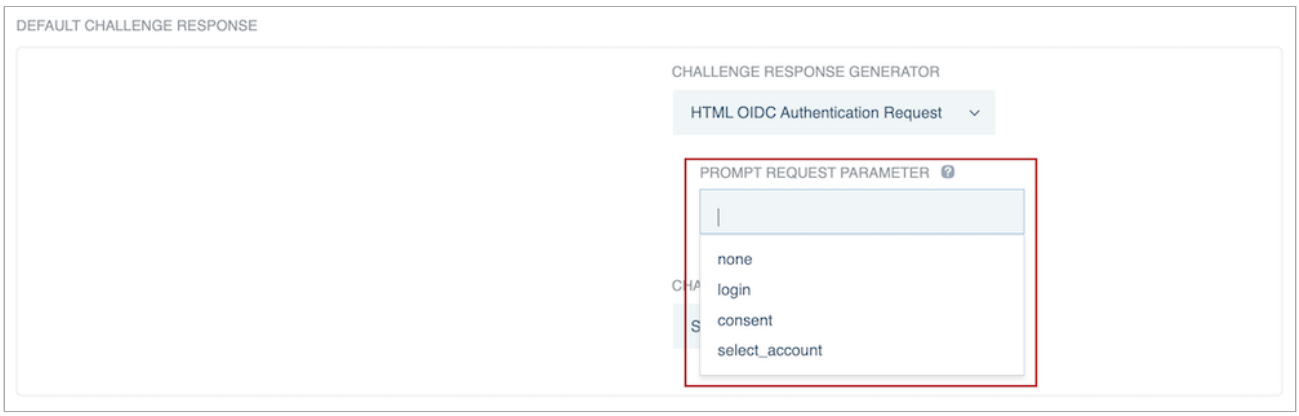
7. **Optional:** To add additional challenge response mappings, click **+ Add Challenge Response Mapping**, then repeat steps 5 and 6.
8. In the **Default Challenge Response** section, select a default challenge response.

   PingAccess uses this challenge response if no other challenge responses apply.

   1. In the **Challenge Response Generator** list, select a challenge response generator.



2. **Optional:** If you selected **Browser-handled OIDC Authentication Request**, **HTML OIDC Authentication Request**, **MS-OFBA Authentication Request Redirect**, **OIDC Authentication Request Redirect**, or **PingFederate Authentication API Challenge**, you can select one of the following options from the **Prompt Request Parameter** list to let the authorization server know whether to prompt an end-user to reauthenticate or provide consent.



none

Returns an error if the end-user isn't authenticated or if the OAuth client doesn't have user consent for the requested claims. The authorization server doesn't prompt the end-user with a consent or authentication page if this option is selected.

login

The authorization server prompts the end-user to reauthenticate. If the end-user doesn't reauthenticate successfully, it returns an error.

> **Note**
>
> For extra security, PingAccess validates the <*auth_time*> the login request was sent at against the <*auth_time*> the OP sets in the response.

consent

The authorization server prompts the end-user for consent before giving information to the OAuth client. If the end-user doesn't give their consent, it returns an error.

select_account

The authorization server prompts the end-user to specify which account they are using, in case they have multiple accounts. If the end-user doesn't select an account, it returns an error.

If you are using PingFederate as the OP, you should also enable push authorization requests on the web session you want to use with this authentication challenge policy. This advanced setting provides an additional layer of security against frontchannel tampering. For more information, see **Enable Push Authorization** in Creating web sessions.

> **Note**
>
> You can set the **Prompt Request Parameter** in two places: on the web session or on one of the OIDC authentication challenge response generators. A value set on a specific authentication challenge response generator takes precedence over one set on a web session.

3. **Optional:** If you selected **OIDC Authentication Request Redirect**, **Redirect Challenge**, or **Templated Challenge**, you can configure PingAccess to let the authentication authority know why a user was redirected to it.
    1. Go to the **Web Sessions** page and expand the web session that you want to edit.

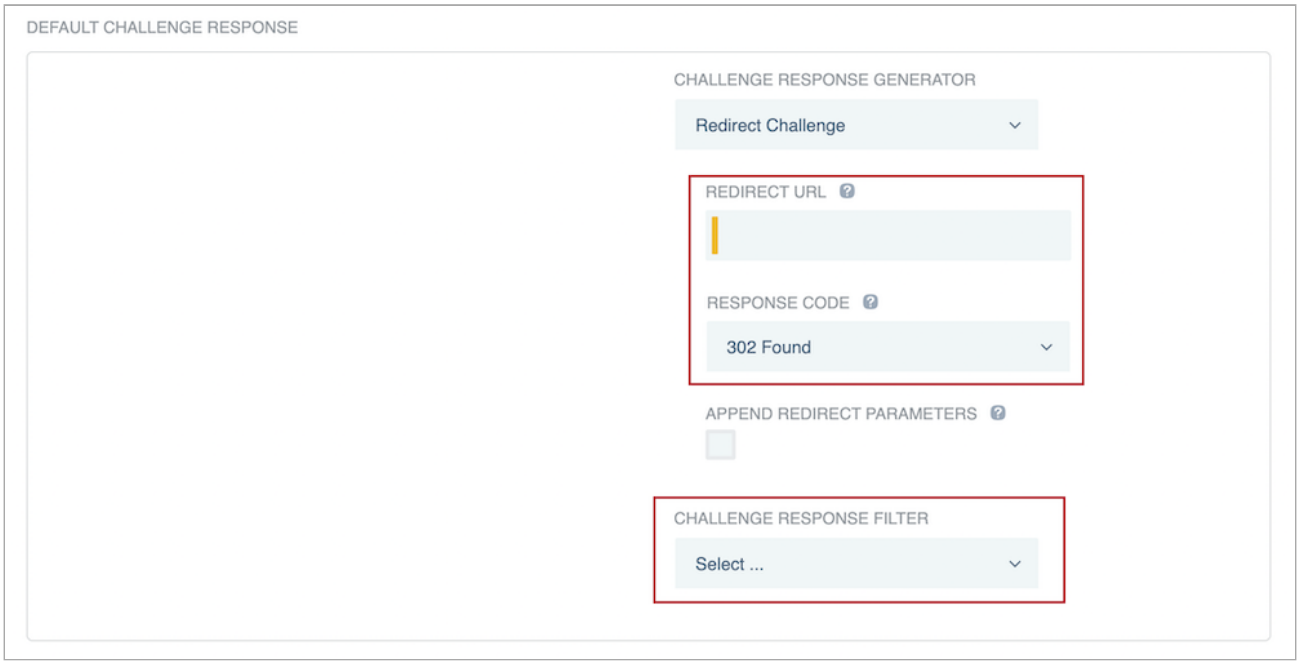
        > **Important**
        >
        > If you selected **Redirect Challenge**, make sure that you select the **Append Redirect Parameters** check box in step 8d.
        > If you selected **Templated Challenge** and are using PingFederate as an authentication source, make sure that you connect to PingFederate's redirectless OIDC flow.

    2. Click the **Pencil** icon, and then select the **Provide Authentication Feedback** check box under **Advanced Settings**.

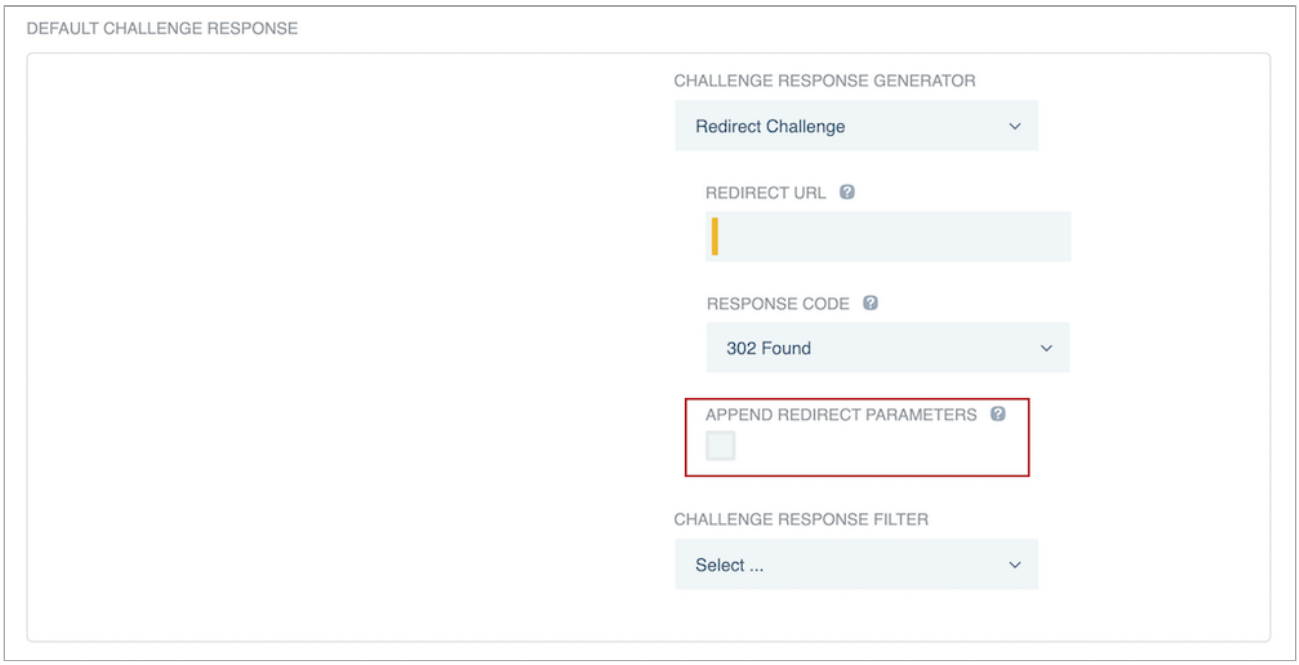
        For more information about the feedback PingAccess can provide, see Creating web sessions.

    An **OIDC Authentication Request Redirect** issues the feedback key `vnd_pi_authn_feedback`. The feedback key issued by an **Redirect Challenge** is `authnFeedback`, and the feedback key issued by an **Templated Challenge** response generator is `oidc.authnFeedback`.

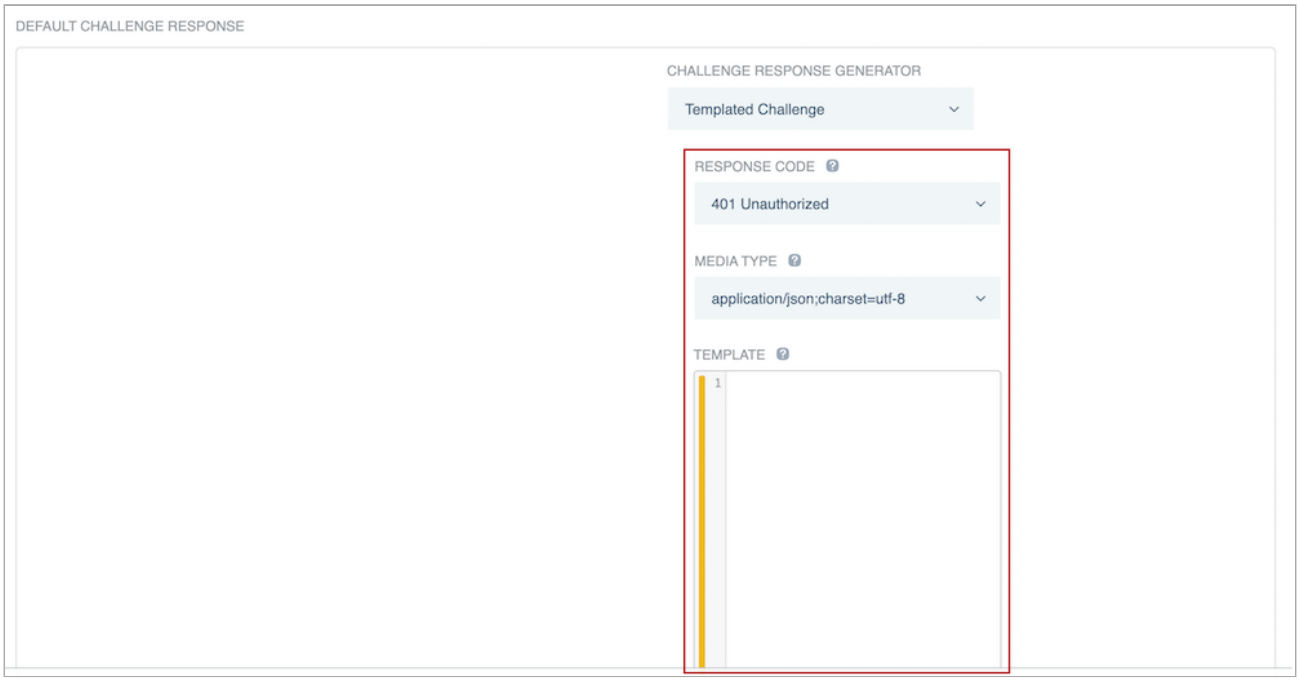
4. If you selected **Redirect Challenge**, enter a **Redirect URL** and select a **Response Code** for the redirect.



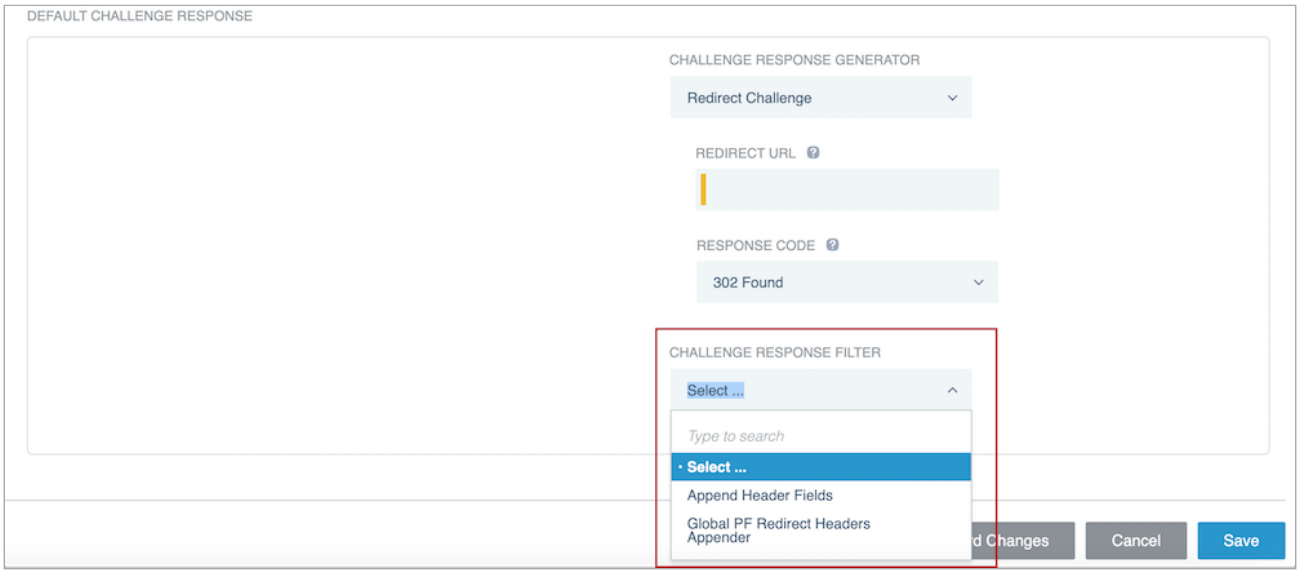
Optionally, select the **Append Redirect Parameters** check box to append PingFederate OIDC parameters and the URL of a requested resource within the query string of a redirect URL that you specify. For more information, see the **Redirect Challenge** table entry in Authentication challenge response generator descriptions.



5. If you selected **Templated Challenge**, select a **Response Code** and **Media Type** for the template, and then enter the template in the **Template** field.



6. In the **Challenge Response Filter** list, select a challenge response filter.



   - If you selected **Append Header Fields**, click **+ Add Row**, then enter a **Name** and **Value** in each row.

Result:

The specified HTTP response header fields are appended to the authentication challenge response.

9. Click **Save**.

Authentication challenge response generator descriptions

This table describes the challenge response generators available for configuration on the **New Authentication Challenge Policy** page.

Challenge Response Generator	Description
Browser-handled OIDC Authentication Request	Generates an HTML or `302` redirect response as described by the Authentication challenge responses tables when single-page application (SPA) support is disabled.
HTML OIDC Authentication Request	Generates a response with a 401 response code. The response body is an HTML document that automatically issues the OpenID Connect (OIDC) authentication request using JavaScript. The HTML always attempts to preserve the fragment of the current browser Uniform Resource Locator (URL) and preserves a POST body if the `Content-Type` is `application/x-www-url-formencoded`.
MS-OFBA Authentication Request Redirect	Adds two response headers to an HTTP request: • `X-FORMS_BASED_AUTH_REQUIRED` • `X-FORMS_BASED_AUTH_RETURN_URL` This enables you to open Microsoft (MS) Office documents protected by PingAccess in an in-app browser that redirects to the OpenID Provider (OP) for user authentication. After the user authenticates, PingAccess establishes a web session and redirects the user to the corresponding MS Office application (spreadsheets open in Microsoft Excel, for example). **Important** This response generator doesn't work with MS Office applications running on macOS, as the macOS in-app browser is much more restrictive. It can't set the nonce cookie that PingAccess requires before redirecting a user. Additionally, Internet Explorer configurations can dictate the behavior of the in-app browser in some environments. If the document you requested fails to download, ensure that **Do not save encrypted pages to disk** is disabled in **Internet Explorer → Internet Options → Advanced → Settings → Security**. **Tip** PingAccess provides an MS-OFBA authentication challenge policy that's included with the system by default. As such, this challenge response generator is best used to address edge cases. For more information, see Authentication.
OIDC Authentication Request Redirect	Generates a response with a 302 response code. The response body directs the browser to send an OIDC authentication request to the OP.

Challenge Response Generator	Description
PingFederate Authentication API Challenge	Generates a response with a 401 response code. The body is a JavaScript Object Notation (JSON) object that directs the application to connect to the PingFederate redirectless authorization application programming interface (API). The JSON object contains three strings: ***authorizationUrl*** Represents the OIDC authentication request. ***method*** Indicates the HTTP method for the request to the PingAccess OIDC callback endpoint. ***oidcAuthnResponseEndpoint*** The location of the PingAccess OIDC callback endpoint. For more information about the required PingFederate configuration, see PingFederate authentication API in the PingFederate documentation. For more information about configuring the JavaScript widget to enable this challenge response, see the Redirectless support page on Github.

Redirect Challenge

Generates a response with the specified response code that redirects the user to a specified URL.

> **Tip**
> To opt out of automatic URL encoding, deselect the **Encode URL** check box. Learn more in PingAccess 8.1 (June 2024).

Optionally, select the **Append Redirect Parameters** check box to append PingFederate Authentication API parameters and the URL of the protected resource the user tried to access within the query string of the redirect URL that you specified.
This lets you initiate PingFederate's redirectless OIDC flow from your own sign-on page when an unauthenticated user tries to access a protected resource. The appended parameters are:

authzUrl
: The OIDC authentication request, similar to `authorizationUrl` from the **PingFederate Authentication API Challenge** response generator. Contains parameters necessary to access the requested resource, such as specific OIDC scopes.

authnResponseMethod
: The HTTP method used to interact with the PingAccess callback endpoint, such as `GET`. resourceUrl
The URL of the resource requested by the user, such as `https://localhost:3000`.

authnResponseEndpoint
: The PingAccess callback endpoint, such as `https://localhost:3000/pa/oidc/cb`.

Challenge Response Generator	Description
	Important When **Append Redirect Parameters** is selected, PingAccess provides the information necessary to complete an OIDC flow within the redirect URL's query string, but it does not automatically redirect the user to the PingFederate authorization endpoint. As such, this setting is best used in conjunction with the redirectless PingFederate authentication API, which reports the current state of an end-user's PingFederate authentication policy flow so that an external web application can manage authentication requests. Regardless of whether you use the Authentication API, you must send a request to the `authzUrl` to start a redirectless sign-on flow with the credentials entered into your sign-on form. This endpoint returns an OIDC token, which you must send to the `authnResponseEndpoint` using the `authnResponseMethod` so that PingAccess can establish a session with the protected resource. After the session is established, you must redirect the user to the `resourceUrl`.

Challenge Response Generator	Description
Templated Challenge	Generates a response with the specified response code based on a specified template. Possible template variables include: • *<resource.name>* represents a string containing the name of the requested resource. • *<application.name>* represents a string containing the name of the requested application. • *<application.realm>* represents a string containing the OAuth realm associated with the application. If the realm is not defined by the application, it is inferred to be the requested authority and the application's context root. • *<exchangeId>* represents a string containing the ID for the current transaction. • *<oidc.authzUrl>* represents the PingFederate OIDC authentication request. Contains parameters necessary to access the requested resource, such as specific OIDC scopes. **Note** Use this variable with the following three variables to initiate PingFederate's redirectless OIDC flow from your own sign-on page when an unauthenticated user tries to access a protected resource, as described in the **Redirect Challenge** response table entry. • *<oidc.authnResponseMethod>* represents the HTTP method used to interact with the PingAccess callback endpoint, such as `GET`. • *<resource.url>* represents the URL of the resource requested by the user, such as `https://localhost:3000`. • *<oidc.authnResponseEndpoint>* represents the PingAccess callback endpoint, such as `https://localhost:3000/pa/oidc/cb`.

Editing authentication challenge policies

Edit the properties of an existing authentication challenge policy in PingAccess.

Steps

1. Click **Access**, then go to **Authentication > Authentication Challenge Policies**.
2. Click to expand the authentication challenge policy you want to edit.

3. Click the **Pencil** icon.
4. Make the desired edits to the authentication challenge policy.
5. To confirm your changes, click **Save**.

Deleting authentication challenge policies

Delete an existing authentication requirements list in PingAccess.

Steps

1. Click **Access**, then go to **Authentication > Authentication Challenge Policies**.
2. Click to expand the authentication challenge policy you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Configuring authentication requirements lists

Configure a list of authentication requirements in PingAccess.

Steps

1. Click **Access**, then go to **Authentication > Authentication Requirements**.
2. Click **+ Add Authentication Requirement**.
3. In the **Name** field, enter a unique name for the authentication requirements list.
4. In the **Authentication Requirements** field, enter an authentication method, such as `cert` or `password`.

   The values you enter here must match the result values defined for the Requested AuthN Context Selector configured within PingFederate. For more information, see Configuring the Requested AuthN Context Authentication Selector.
5. To add one or more additional authentication requirements, click **+ Add Authentication Requirement**.
6. Click **Save**.

Editing authentication requirements lists

Edit the properties of an existing authentication requirements list in PingAccess.

Steps

1. Click **Access**, then go to **Authentication > Authentication Requirements**.
2. Click to expand the list you want to edit.
3. Click the **Pencil** icon.

4. Make the desired edits to the authentication requirements list.
5. To confirm your changes, click **Save**.

Deleting authentication requirements lists

Delete an existing authentication requirements list in PingAccess.

Steps

1. Click **Access**, then go to **Authentication > Authentication Requirements**.
2. Click to expand the authentication requirement you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Identity mappings

Identity mappings make user attributes available to backend sites that use them for authentication.

There are multiple types of identity mappings, each with a different behavior and a distinct set of fields to specify the identity mapping behavior. Learn more about what you can do with identity mappings in:

- Creating header identity mappings
- Creating JWT identity mappings
- Creating web session access token identity mappings

Learn more about managing existing identity mappings in:

- Editing identity mappings
- Deleting identity mappings

For JSON Web Token (JWT) identity mappings, you can find more information about how to set up the issuer and signing configuration in:

- Configuring auth token management

Creating header identity mappings

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 three levels deep. Header identity mappings are assigned to applications.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the mapping.
4. In the **Type** list, select **Header Identity Mapping**.
5. In the `Attributes` section, select a list type.

   Choose from:

   - **Inclusion List**: Includes the specified attributes as headers.
   - **Exclusion List**: Includes all attributes as headers, with the exception of those specified.
6. If you selected **Inclusion List**, specify the **Inclusion List** parameters.
   1. In the **Header Name Prefix** field, enter a prefix.

      This prefix is prepended onto all header names.
   2. In the **Attribute Name** field, enter or select the name of the attribute to retrieve from the user web session in the **Attribute Name** field.

      For example, `sub`.
   3. In the **Header Name** field, enter the name of the HTTP requests header to contain the attribute value.

      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`.
   4. **Optional:** Click **+ Add Row** to add additional sets of attributes and headers.
   5. **Optional:** Click **Subject** to select which attribute is used as the subject.
7. If you selected **Exclusion List**, specify the **Exclusion List** parameters.
   1. In the **Header Name Prefix** field, enter a prefix.

      This prefix is prepended onto all header names.
   2. **Optional:** In the **Excluded Attributes** field, enter one or more attributes to exclude.

      All attributes not specified are included as headers.
   3. In the **Subject Attribute Name** list, select an attribute to use as the subject.
8. In the **Certificate to Header Mapping** section, enter the header name included in 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.
   1. If you are using a certificate chain, click **+ Add Row** to add another row.
9. Click **Save**.

Creating JWT identity mappings

To make user attributes available in a signed JSON Web Token (JWT) sent to the application using a header, create a JWT identity mapping.

Before you begin

Define the JWT issuer and signing configuration in Configuring auth token management.

About this task

When configuring identity mappings, PingAccess supports dot notation to maintain session token structure. 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 entries in the following table 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

Backend sites can use the JWKS (JSON Web Key Set) endpoint available at `/pa/authtoken/JWKS` to validate the JWT signature. Learn more about the endpoint in OpenID Connect endpoints.

Custom claim values can be:

- Claims obtained from the user identity
- Dot expressions
- Literal values such as a string, boolean, integer, or array

> **Tip**
>
> Use a literal value when you don't expect an identity mapping to be present for the user, such as for anonymous users.

To define multiple values for a claim, separate the values with a dot. For example, `value1.value2`.

To separate multiple dot expressions, you must use a `/` character to escape any subsequent values. For example, `pi/.pa/.rat`.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a unique name for the mapping.
4. In the **Type** list, select **JSON Web Token (JWT)**.
5. To map the identity JWT as a bearer token in the Authorization request header, select **Map as Bearer Token**.

   If you select this option, the JWT token replaces any existing tokens and the **Header Name** field isn't usable in this identity mapping configuration.
6. In the **Header Name** field, enter the name of the header you want to use when sending the signed JWT to the target application.

   Configure the actual header name over the HTTP protocol, and don't use an environment variable interpreted format.

   For example, if you're using the `User-Agent` header to identify browser types, enter it as `User-Agent`, not `HTTP_USER_AGENT`.
7. In the **Audience** field, enter the audience you want to set as the `aud` claim in the signed JWT.
8. In the **JWT Claim Mapping** section, select a list type:

   Inclusion List

   Uses only the attributes you specify.

   Exclusion List

   Uses all attributes except for the ones you specify.
9. If you selected **Inclusion List**, configure a **JWT Claim Mapping**:
   1. Click **+ Add Row**.
   2. In the **User Attribute Name** field, enter or select the name of the attribute to retrieve from the user web session, such as `sub_jwk`.
   3. In the **JWT Claim Name** field, enter the name of the JWT claim to contain the attribute value.
   4. Click **+ Add Row** and repeat as necessary.
   5. Select which included attribute to use as the **Subject**.

10. If you selected **Exclusion List**:
    1. In the **Excluded Attributes** field, enter the names of any attributes that you don't want to collect from the user web session.
    2. In the **Subject Attribute Name** field, enter or select the name of an included attribute you want to use as the subject.
11. (Optional) In the **Client Certificate Chain JWT Claim Name** field, enter the name of the JWT claim to contain the client certificate chain array.
    1. If you are performing Certificate to JWT Claim Mapping, in the **Client Certificate Chain Max Depth** field, enter the maximum number of certificates from the client certificate chain included in the JWT claim array.
12. To configure advanced settings, click **Show Advanced**.
    1. (Optional) To use a cached signed JWT for repeated requests for a given user, click **Show Advanced** and select **Cache JWT**.

       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.
    2. In the **Custom JWT Claims** section, configure any custom claims you want to add to the JWT token:
       1. Click **+ Add Row**.
       2. In the **Name** field, enter the name of the JWT claim to contain the attribute value.

          > **Important**
          >
          > If you configure a custom JWT claim with a claim name that you've already used in the inclusion or exclusion list, the following applies:
          > - Custom claims don't override attributes in the **Exclusion List**.
          > - If a claim in the **Inclusion List** results in a null value, PingAccess uses the custom value instead.
          > - If the **Inclusion List** overrides the custom value, PingAccess logs a debug message indicating that the custom value wasn't used.

       3. In the **Value** field, enter the name of the attribute you want to retrieve from the user web session.
       4. Click **+ Add Row** and repeat as necessary.
    3. To create a JWT containing only the custom claims defined in step 12b when identity information is unavailable, select **Create JWT for Anonymous Users**.

       This checkbox enables you to use JWT identity mapping to send context that might not require a user identity.
    4. In the **Type Header Value** field, enter a value to use as the `typ` header in the JWT, indicating what type of JWT it is.

       > **Note**
       >
       > To comply with IETF RFC 9068 - JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens when using a JWT access token, enter the recommended value of `at+jwt`. If `application/at+jwt` is required, you can use this value instead.

13. Click **Save**.

Creating web session access token identity mappings

Add a web session access token identity mapping to PingAccess.

About this task

PingAccess can map the access token from a web session to the bearer authentication token of request headers. This lets PingAccess integrate with protected APIs that expect OAuth access tokens issued by the same OAuth authorization server used in PingAccess.

Steps

1. In the administrative console, click **Access** then go to **Identity Mappings → Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name for the new identity mapping.
4. From the **Type** list, select **Web Session Access Token Identity Mapping**.
5. Click **Save**.

Editing identity mappings

Edit the properties of an existing identity mapping in PingAccess.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click to expand the identity mapping you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the identity mapping.
5. To confirm your changes, click **Save**.

Deleting identity mappings

Delete an existing identity mapping in PingAccess.

Steps

1. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click to expand the identity mapping you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Configuring auth token management

To define the issuer and signing configuration used by JSON Web Token (JWT) identity mappings, configure auth token management.

Steps

1. Click **Access**, then go to **Identity Mappings > Auth Token Management**.
2. To enable key rolling using the specified key roll interval, click **Key Roll Enabled**.
3. To indicate how often, in hours, you want to roll the keys, specify the **Key Roll Interval (h)**.

   Key rollover updates keys at regular intervals to ensure the security of the signed auth tokens.
4. In the **Issuer** field, specify a published, unique issuer identifier to use with auth tokens.

   Set the issuer to a value that more closely represents your company. PingAccess inserts this value as the iss claim within the auth token.
5. In the **Signing Algorithm** field, select the signing algorithm used to protect the integrity of the auth tokens.

   The default is ECDSA using P-256 Curve.
6. Click **Save**.

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 tasks within this section 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 PingAccess 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 PingAccess 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 PingAccess token is sent.

> **Note**
>
> 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 Uniform Resource Locator (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.

Configuring web session management settings

Configure web session management settings in PingAccess.

Steps

1. Click **Access**, then go to **Web Sessions > Web Session Management**.
2. In the **Web Session Management** section, select **Key Roll Enabled** to enable key rolling using the interval specified below.
3. 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 PingAccess tokens.
4. In the **Issuer** field, enter the published, unique identifier to be used with the web session (the default is PingAccess).

   Example:

   Set the issuer to a value that more closely represents your company. PingAccess inserts this value as the `iss` claim within the PingAccess token
5. Select the **Signing Algorithm** used to protect the integrity of the PingAccess token (the default is `ECDSA using P-256 Curve`).

   PingAccess uses the algorithm when creating signed PingAccess 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 PingAccess tokens are encrypted
6. Select the **Encryption Algorithm** used to encrypt and protect the integrity of the PingAccess Token (the default is `AES 128 with CBC and HMAC SHA 256`).

   PingAccess uses the algorithm when creating encrypted PingAccess tokens and when verifying them from a user's browser.

   Higher encryption levels are available if the administrative console supports it. To enable higher encryption levels, update the administrative console Java Runtime Environment (JRE) to support unlimited strength security policy.

   In a clustered environment, add the security policy changes to the engines as well as the administrative console for the cluster.

7. Enter the browser **Cookie Name** that contains the PingAccess token (the default is `PA`).
8. In the **Session State Cookie Name** field, enter a name for the browser cookie to contain session state attributes.
9. In the **Login State Cookie Name** field, enter a name for the browser cookie that contains the sign-on state.

   This temporary cookie tracks the number of sign-on reattempts that occur if you've configured a value for **Max Login Retries** in Advanced web session settings. PingAccess clears the cookie after a sign-on attempt succeeds.

   Applicable only when the **OpenID Connect Login Type** is **Code**. Learn more in step 6 of Creating web sessions.

   The default value is `PA_LOGIN`.

   > **Note**
   >
   > Make sure to configure different values for **Cookie Name**, **Session State Cookie Name**, and **Login State Cookie Name**.

10. In the **Update Token Window (s)** field, enter the number of seconds before the idle timeout is updated in the PingAccess token.

    When this time window expires, PingAccess will reissue a new PingAccess cookie.

11. 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.

12. In the **Nonce SameSite Cookie** list, select a level of restriction for when cookies can be sent in a cross-site request:

    Choose from:

    - **Lax**: The cookie should be sent on initial navigation to a site. It can be sent in same-site requests but not cross-site requests.
    - **Strict**: The cookie can't be sent in top-level cross-site requests.

      > **Note**
      >
      > The `SameSite=Strict` attribute provides greater protection against cross-site request forgery (CSRF), but cannot fully prevent it. Use the `SameSite=Strict` attribute as part of a more comprehensive CSRF protection strategy. Learn more in https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-rfc6265bis-14#section-8.8.

    - **None**: The cookie can be used across different sites without restriction.

      > **Note**
      >
      > To prevent browser compatibility issues, if PingAccess detects that the user's browser matches any of the values set in the `pa.websession.cookie.sameSiteExcludedUserAgentPatterns` property in the `run.properties` file, PingAccess doesn't add the `SameSite=None` attribute to cookies.

    - **Disabled**: PingAccess doesn't set the `SameSite` attribute. The browser determines how to handle the cookie.

> **Note**
>
> A browser issue can prevent sign on if the `SameSite Cookie` attribute is set. Learn more in the PingAccess 7.0 SameSite cookie upgrade issue release note entry.

- **Legacy** (default): Maintain the same behavior as in PingAccess 8.1 and earlier:
  - PingAccess sets the nonce cookie without a `SameSite` setting if either:
    - The web session is set to the **Disabled** `SameSite` setting.
    - The user-agent matches one of the `pa.websession.cookie.sameSiteExcludedUserAgentPatterns`.
  - PingAccess sets the nonce cookie to **SameSite=None** if:
    - The web session is set to any `SameSite` setting other than **Disabled** and does not match one of the `pa.websession.cookie.sameSiteExcludedUserAgentPatterns`.

13. Click **Save**.

Creating web sessions

Create a new web session in PingAccess to define a policy for web application session creation, lifetime, timeouts, and scope. You can apply the same web session to multiple target applications.

Before you begin

If you are using PingFederate as the OpenID Connect (OIDC) token provider and plan to use **Mutual TLS**, you must make two changes to the PingFederate configuration:

1. Enable the use of the secondary HTTPS port in PingFederate by editing the `<PF_HOME>/pingfederate/bin/run.properties` file and setting the `pf.secondary.https.port` property to a port value. Learn more in Configuring PingFederate properties.
2. Edit the `openid-configuration.template.json` file to add the `mtls_endpoint_aliases` object, with content defined by RFC-8705. Learn more about this file in customizing the OpenID provider configuration endpoint response.

Steps

1. Click **Access**, then go to **Web Sessions > Web Sessions**.
2. Click **+ Add Web Session**.
3. In the **Name** field, enter a unique name for the web session.

   The name can be up to 64 characters, including special characters and spaces.
4. In the **Cookie Type** list, select a type of token to create.

   Choose from:

   - **Encrypted JWT** (default): An encrypted JSON Web Token (JWT) uses authenticated encryption to simultaneously provide confidentiality, integrity, and authenticity of the PingAccess token.

- **Signed JWT**: A signed JWT uses asymmetric cryptography with a private-public key pairing to verify the signed message and confirm that it wasn't modified during transit.

> **Note**
>
> Changing this setting could affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

5. In the **Audience** field, enter a short, unique identifier between 1 - 32 characters to define the audience to which the PingAccess token applies.

   Requests made to a target application that's associated with this web session must have a PingAccess token that matches the configured audience value. Otherwise, PingAccess redirects to the OIDC provider.

> **Note**
>
> Changing this setting can affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

6. In the **OpenID Connect Login Type** list, select a method of verifying the user and collecting additional profile claims.

   OIDC supports three login types that define how the user's identity is verified based on authentication performed by an OpenID Provider, and how additional profile claims are obtained. Three OIDC sign-on profiles are supported: **Code**, **POST**, and **x_post**.

   Choose from:

   - **Code**: A standard OIDC login flow that provides confidentiality for sensitive user claims. In this profile the relying party, PingAccess, makes multiple back-channel requests to exchange an authorization code for an ID token. PingAccess then exchanges an access token for additional profile claims from the **UserInfo** endpoint at the provider, PingFederate. Use this login type 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 and requires PingFederate 7.3 or later.

     After authentication, PingFederate sends a form auto-POST response containing the ID token, including profile claims, to PingAccess through the browser. Backchannel communication between PingAccess and PingFederate is required for key management to validate ID tokens. If the exchanged claims do not contain information that should be hidden from the end user, use this login type for maximum performance.

     Select the **Implicit** grant type when configuring the OAuth Client within PingFederate. Learn more in Configuring OAuth clients. You must set the ID token-signing algorithm in PingFederate to one of the ECDSA or RSA algorithms.
   - **x_post**: A login flow based on OIDC that passes claims from the provider through the browser. Select the **Implicit** grant type and use one of the ECDSA or RSA algorithms as the ID token-signing algorithm.

> **Note**
>
> If you are using PingFederate 7.3 or later, use `POST` rather than `x_post`, which Ping Identity defined prior to the development of the OAuth 2.0 Form Post Response Mode draft specification.

7. In the **Client ID** field, select the unique identifier that you were assigned when creating the OAuth relying party client within the OpenID provider.

If PingFederate is the OpenID provider, learn more in Configuring OAuth clients.

8. In the **Client Credentials Type** section, click the type of credential that you were assigned when you created the OAuth relying party client within the OpenID provider, then provide the corresponding information.

   > **Important**
   >
   > If you enable session validation or configure the **Code** login type, this field is required.

   Choose from:

   - **Secret**: Enter the **Client Secret** that was assigned when you created the OAuth relying party client within the token provider.
   - **Mutual TLS**: Select a configured **Key Pair** to use for Mutual TLS client authentication.
   - **Private Key JWT**: If you selected the **Enable Static Keys** check box on the **Static OAuth & OpenID Connect Keys** page, you must select an active **Signing Algorithm**.

     If you have not selected **Enable Static Keys**, the **Signing Algorithm** field is optional and defaults to the dynamic signing algorithm set on the **OAuth Key Management** page. Learn more in Configuring static signing keys.

   > **Note**
   >
   > If PingFederate is the OpenID provider, the OAuth client that you use with PingAccess web sessions must have an OIDC policy specified. Learn more in Configuring OpenID Connect Policies.

9. In the **Idle Timeout** field, enter the amount of time, in minutes, that the PingAccess token remains active if PingAccess doesn't detect any user activity.

   The default value is `60` minutes. If an idle timeout occurs, PingAccess automatically terminates the associated session.

   > **Note**
   >
   > If the user has a valid existing PingFederate session when an idle timeout occurs in an associated PingAccess session, the PingAccess session might re-establish itself without prompting the user to sign on again.

10. In the **Max Timeout** field, enter a maximum amount of time, in minutes, that the PingAccess token remains active.

    The default value is `240` minutes. When the PingAccess token expires, the associated user must reauthenticate. This protects against unauthorized resource use by ensuring that sessions end by the specified time and require the associated user to reauthenticate to continue.

    > **Note**
    >
    > If PingFederate is the token provider, this value must be smaller than the PingFederate access token lifetime defined in the PingFederate access token management instance. Learn more in Reference token management.

11. **Optional:** To configure advanced settings, click **Show Advanced**.

    Learn more about the configuration options in Configuring advanced web session settings.

12. Click **Save**.

Configuring advanced web session settings

Configure advanced settings on a web session for more control over how to set cookies, preserve requests, and more. These settings are optional.

Steps

- In the **Cookie Domain** field, enter a valid domain where the cookie is stored, such as `corp.yourcompany.com`.

  If you set a cookie domain, all of your web resources must reside within that domain. If you do not set a cookie domain, the PingAccess token is recreated for each host domain where you access applications.

  > **Important**
  >
  > Setting an invalid **Cookie Domain** causes authentication to fail. This results in PingAccess redirecting the user to reauthenticate with PingFederate indefinitely.

- Select **Secure Cookie** to indicate that the PingAccess cookie must be sent through HTTPS connections only.

  This checkbox is selected by default.

  > **Important**
  >
  > Selecting **Secure Cookie** in a non-HTTPS environment causes authentication to fail. This results in PingAccess redirecting the user to reauthenticate with PingFederate indefinitely.

- Select **HTTP-Only Cookie** to enable the `HttpOnly` flag on cookies that contain the PingAccess token.

  This checkbox is selected by default.

  Cookies that have this flag aren't accessible through the use of non-HTTP methods such as JavaScript calls (for example, referencing `document.cookie`), and cannot be easily stolen using cross-site scripting.

- Select **Partitioned Cookie** to add the `Partitioned` attribute to all cookies that PingAccess sets for this web session.

  This checkbox is cleared by default.

  Selecting **Partitioned Cookie** ensures that cross-site cookies will continue to be readable within the same context that they're created in. Learn more in PingAccess 8.1 (June 2024).

  > **Tip**
  >
  > Use the **Partitioned Cookie** checkbox to override the value of the `pa.default.session.cookie.attributes.partitioned` property in the `run.properties` file for this web session without needing to apply changes to all of the nodes in a PingAccess cluster and restart them.

- Select **Enable PKCE** to prompt PingAccess to send a SHA256 code challenge and corresponding code verifier as a proof key for code exchange during the code authentication flow.

  This checkbox is selected by default.

- In the **SameSite Cookie** list, select a level of restriction for when cookies can be sent in a cross-site request.

Choose from:

- **Lax** (default): The cookie should be sent on initial navigation to a site. It can be sent in same-site requests but not cross-site requests.
- **Strict**: The cookie can't be sent in top-level cross-site requests.

> **Note**
>
> The `SameSite=Strict` attribute provides greater protection against cross-site request forgery (CSRF), but cannot fully prevent it. Use the `SameSite=Strict` attribute as part of a more comprehensive CSRF protection strategy. Learn more in section 8.8 in IETF RFC 6265 - Cookies: HTTP State Management Mechanism.

- **None**: The cookie can be used across different sites without restriction.

> **Note**
>
> To prevent browser compatibility issues, if PingAccess detects that the user's browser matches any of the values set in the `pa.websession.cookie.sameSiteExcludedUserAgentPatterns` property in the `run.properties` file, PingAccess doesn't add the `SameSite=None` attribute to cookies.

- **Disabled**: PingAccess does not set the `SameSite` attribute. The browser determines how to handle the cookie.

> **Note**
>
> A browser issue can prevent sign on if the `SameSite Cookie` attribute is set. Learn more in the PingAccess 7.0 SameSite cookie upgrade issue release note entry.

- Enter any additional **Scopes** that you want to request from the OIDC token provider during the ID token request.

  If you have a token provider configured, you can select published scopes from the list based on the selected Client ID.

  To select unverified scopes, enter the scope and select **Use unverified scope "[scopename]"**.

> **Note**
>
> You must configure the token provider to handle all the scopes that you select, including any custom scope values.

> **Important**
>
> Users can access all attributes by examining browser traces. Although they are integrity protected to prevent changes, sensitive or confidential attributes can still be viewed if the user decodes the ID Token's value.

- In the **Prompt Parameter** list, select one of the following to let the authorization server know whether to prompt an end user to reauthenticate or provide consent.

  Choose from:

  - **none**: Returns an error if the end user isn't authenticated or if the OAuth client doesn't have user consent for the requested claims. If selected, the authorization server doesn't prompt the end user with a consent or authentication page.

- **login**: The authorization server prompts the end user to reauthenticate. If the end user doesn't reauthenticate successfully, it returns an error.

  > **Note**
  >
  > For extra security, PingAccess validates the *<auth_time>* the login request was sent at against the *<auth_time>* the OpenID Provider (OP) sets in the response.

- **consent**: The authorization server prompts the end user for consent before giving information to the OAuth client. If the end user doesn't consent, it returns an error.

- **select_account**: The authorization server prompts the end user to specify which account they are using. If the end user doesn't select an account, it returns an error.

PingAccess configures the prompt parameter in the OIDC authentication requests that it generates. The token provider uses the value of the prompt parameter to confirm the end user's presence or to emphasize the authentication request's importance.

> **Important**
>
> You can set the **Prompt Parameter** on both the web session and on one of the OIDC authentication challenge response generators. A prompt parameter value set on a specific authentication challenge response generator takes precedence over one set on a web session. Learn more in Configuring authentication challenge policies. Select the **Enable Push Authorization** advanced setting if you're using PingFederate as a token provider. This setting provides an extra layer of security against frontchannel tampering.

- Select **Enable Push Authorization** to have PingAccess send a backchannel request at the token provider's pushed authorization request (PAR) endpoint.

  PingAccess uses the token provider's metadata value for the `pushed_authorization_request_endpoint` to determine where to send the request.

  This checkbox is cleared by default.

  > **Important**
  >
  > Currently, PingAccess only supports PingFederate as the OP for PAR. PingAccess 7.2 and later use the OAuth 2.0 Pushed Authorization Requests (PAR) draft specification.
  > PingFederate added support for this specification in version 10.2. Learn more in the PingFederate 10.2 release notes.

- In the **Validate Session** section, click **Yes** to validate PingAccess sessions with a configured PingFederate instance during request processing.

  No is selected by default.

  If you enable this setting, PingAccess synchronizes with session logouts from PingFederate.

  > **Important**
  >
  > To use this setting, you must perform additional configuration steps in PingFederate. Learn more in Server-side session management configuration.

> **Note**
>
> Changing this setting might affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

1. In the **PingFederate Session State Cache** field, enter a maximum amount of time, in seconds, before session synchronization takes effect.

   The default value is `60` seconds.

   > **Note**
   >
   > This setting is only available if you enable the **Validate Session** setting.

- In the **Refresh User Attributes** section, click **Yes** to have PingAccess periodically contact PingFederate to update user data that's used in evaluating policy claims. Otherwise, click **No**.

  Yes is selected by default.

  User data refreshes occur when processing a request from the user's session that has surpassed the **Refresh User Attributes Interval**. PingAccess caches the user attributes until the **Refresh User Attributes Interval** is surpassed again. This continues until the user session is terminated.

  This option works in conjunction with the PingAccess web session management features to automatically require user reauthentication if the user attribute data used as issuance criteria for a token in PingFederate causes the token to be revoked.

  PingFederate provides data according to its OIDC policy, which can pull data from the access token or from an attribute source lookup:

  - If it pulls data from the access token, the data doesn't change until the token expires.
  - If it pulls data from an attribute source lookup, the new data is available whenever the query is made.

  > **Note**
  >
  > If the PingFederate OIDC policy uses an attribute source lookup and has issuance criteria configured to only issue a token if the account is enabled, enabling **Refresh User Attributes** allows PingAccess to terminate the session the next time that 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 that user data is cached, so a change that terminates a session can take up to 60 seconds (by default) to take effect.

  > **Note**
  >
  > Changing this setting can affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

  1. In the **Refresh User Attributes Interval** field, enter a maximum amount of time, in seconds, that PingAccess caches user data.

     The default value is `60` seconds.

> **Note**
>
> This setting is only available if you enable the **Refresh User Attributes** setting.

- Select **Cache User Attributes** to have PingAccess cache user attributes internally for use in policy decisions:

  This checkbox is cleared by default.

  - If you enable this setting, an attribute list that's longer than the maximum cookie size can contain information used to evaluate access requests. Maximum cookie size is typically `4096` bytes.
  - If you don't enable this setting, user attribute data is encoded, signed, or encrypted, depending on the web session cookie type, then stored in the browser's cookie store. The information is sent from the browser back to PingAccess with each request.

    > **Note**
    >
    > Changing this setting can affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

- In the **Request Preservation** list, select a type of request data for PingAccess to preserve if the user is redirected to an authentication page when submitting information to a protected resource.

  Choose from:

  - **None**: PingAccess doesn't preserve any request data.
  - **POST** (default): PingAccess preserves POST data only.
  - **POST and Fragment**: PingAccess preserves both POST and fragment data.

- In the **Web Storage** list, select the type of web storage that PingAccess uses to temporarily preserve data about non-authenticated requests in the browser.

  Choose from:

  - **Session Storage** (default): PingAccess stores request preservation data for the current session. After the user authenticates successfully or closes the associated tab, PingAccess clears the data.
  - **Local Storage**: PingAccess stores request preservation data across multiple sessions on the same device. After the user authenticates successfully, PingAccess clears the data.

    > **Note**
    >
    > If users commonly use Internet Explorer with security zones enabled and PingFederate is in a different zone than PingAccess, use **Local Storage**.

- In the **Max Login Retries** field, enter the maximum number of times PingAccess should retry a sign-on attempt after an authorization code exchange failure.

  To allow infinite retries, enter a value of `0`. To prevent PingAccess from retrying sign-on attempts, enter `-1`. The default value is `-1`.

> **Note**
>
> Applicable only when the **OpenID Connect Login Type** is **Code**. Learn more in step 6 of Creating web sessions.

- In the **Login Retry Delay (Sec.)** field, enter the number of seconds PingAccess should wait before retrying a sign-on attempt after an authorization code exchange failure.

  The default value is `0`.

> **Note**
>
> Applicable only when the **OpenID Connect Login Type** is **Code** and a non-negative value is configured in the **Max Login Retries** field. Learn more in step 6 of Creating web sessions and in the previous step.

- In the **Timeout Groovy Script** field, enter a Groovy script to customize web session durations for different types of users based on identity attributes that the token provider returns.

  Create a **Timeout Groovy Script** to overwrite a web session's default *<sessionTimeout>* (representing the **Max Timeout** field) and *<idleTimeout>* (representing the **Idle Timeout** field) values. The following methods are unique to the **Timeout Groovy Script**:

  ### *createTimeout()*

  Creates an empty SessionTimeoutContext object.

  ### *createTimeoutIdle(x)*

  Creates an object with an *<idleTimeout>* of x minutes.

  ### *createTimeoutSession(x)*

  creates an object with a *<sessionTimeout>* of x minutes.

  ### *createTimeout(x,y)*

  Creates an object with a *<idleTimeout>* of x minutes and a *<sessionTimeout>*of y minutes.

  Enter specific user attributes in the script, such as `client_id` in the following example script:

```
def context = createTimeout();
if (user.has("client_id"))
 {
   context = createTimeoutIdle(15);
 }
return context;
```

  When a user with one of the specified attributes opens a web session, PingAccess modifies the duration of the session according to the values that you set.

> **Note**
>
> When PingAccess issues a PingAccess token, it packages user attributes into a JsonNode object. Learn more about methods that might be useful, such as `JsonNode get()` or `boolean has()`, in the JsonNode object reference.

- Select the **Provide Authentication Feedback** checkbox to configure PingAccess to let the authentication authority know why a user was redirected to it.

  This checkbox is cleared by default.

  Currently, PingAccess only provides this challenge reason feedback if the user's web session times out.

  This option is available for the **OIDC Authentication Request Redirect**, **Redirect Challenge**, and **Templated Challenge** response generators. Learn more about the next steps in step 6d in the Configuring authentication challenge policies.

  > **Important**
  >
  > PingAccess only sends a feedback key to the authentication authority. From there, the authentication authority must configure and display a user-facing message explaining why the redirect was necessary. In PingFederate, for example, the feedback key can be used within a velocity template. Velocity templates let you create custom user-facing messages, such as `Your session has expired. Please log in again.`

  > **Tip**
  >
  > On the web session admin API endpoint, **Provide Authentication Feedback** is expressed as `provideAuthenticationFeedback`.

- In the **Type Header Value** field, enter a value to use as the `typ` header in the JWT, indicating what type of JWT it is.

  > **Note**
  >
  > To comply with IETF RFC 9068 - JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens when using a JWT access token, enter the recommended value of `at+jwt`. If `application/at+jwt` is required, you can use this value instead.

Editing and deleting web sessions

Edit or delete a web session in PingAccess.

About this task

To edit the properties of an existing web session:

Steps

1. Click **Access**, then go to **Web Sessions > Web Sessions**.
2. Click to expand the web session you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits to the web session. Click **Save**.

Deleting web sessions

About this task

Delete an existing web session in PingAccess. If the web session is currently associated with an application, you cannot delete it.

Steps

1. Click **Access**, then go to **Web Sessions > Web Sessions**.
2. Click to expand the web session you want to delete.
3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Token validation

You can configure application programming interface (API) and Web + API applications to use access token validators to locally verify signed and encrypted access tokens. This feature works in conjunction with token providers that support JSON Web Signature (JWS) and JSON Web Encryption (JWE) validation.

> **Tip**
>
> When using PingFederate as the token provider for this feature, export the `Generated: ENGINE` keypair from PingAccess, located under **Security → Key Pairs**, and import to PingFederate trusted certificate authorities (CAs).

Adding access token validators

Add an access token validator to verify signed or encrypted access tokens in PingAccess.

Steps

1. Click **Access**, then go to **Token Validation > Access Token Validators**.
2. Click **+ Add Access Token Validator**.
3. In the **Name** field, enter a name for the token validator.
4. In the **Type** list, select the type of key you want to validate.

   The token provider configuration specifies which type of key. You can find information about configuring PingFederate as the token provider in Configuring JSON token management.
5. (Optional) In the **Description** field, enter a description for the token validator.
6. In the **Path** field, specify the endpoint path to verify the signature.

   This entry must start with a forward slash (/), and must not end with a forward slash (/). PingFederate token provider configuration informs the host and port. PingAccess permits query strings in the path.
7. (Optional) In the **Subject Attribute Name** field, enter the attribute expected as the subject.

If this value is configured and the specified subject attribute name isn't present in the token, validation fails.

8. (Optional) In the **Issuer** field, enter the expected value of the issuer to include in the access token.

   If this value is configured and the specified issuer isn't present in the token, validation fails.

9. (Optional) In the **Audience** field, specify the audience value to include in the access token.

   If this value is configured and the specified audience isn't present in the token, validation fails.

10. If you don't want to validate access tokens for an audience value, you must select the **Skip Audience Validation** checkbox.

11. Click **Save**.

Adding multiple JWKS endpoint access token validators

Add a **Multiple JSON Web Key Set (JWKS) Endpoint** access token validator to define multiple endpoints or issuers.

Steps

1. Click **Access**, then go to **Token Validation > Access Token Validators**.
2. Click **+ Add Access Token Validator**.
3. In the **Name** field, enter a name for the token validator.
4. In the **Type** list, select Select **Multiple JSON Web Key Set (JWKS) Endpoint**.
5. (Optional) In the **Description** field, enter a description for the token validator.
6. In the **Path** field, specify the endpoint path to verify the signature.

   This entry must start with a forward slash (/), and must not end with a forward slash (/). PingFederate token provider configuration informs the host and port. PingAccess permits query strings in the path.

7. (Optional) In the **Subject Attribute Name** field, enter the attribute expected as the subject.

   If this value is configured and the specified subject attribute name isn't present in the token, validation fails.

8. (Optional) In the **Issuer** field, enter the expected value of the issuer to include in the access token.

   If this value is configured and the specified issuer isn't present in the token, validation fails.

   > **Note**
   >
   > A **Multiple JSON Web Key Set (JWKS) Endpoint** access token validator (ATV) processes each JWKS with the matching issuer from the access token. The issuer value is looked at first, if it's present.
   > If a matching issuer isn't configured, the ATV cycles through all the JWKS endpoints until it finds the one that works.

9. (Optional) In the **Audience** field, specify the audience value to include in the access token.

   If this value is configured and the specified audience isn't present in the token, validation fails.

10. If you don't want to validate access tokens for an audience value, you must select the **Skip Audience Validation** checkbox.

11. Click **+ Add Row** and repeat steps 6 - 10 for any additional endpoints.

12. Click **Save**.

Editing access token validators

Edit an existing access token validator in PingAccess.

Steps

1. Click **Access**, then go to **Token Validation > Access Token Validators**.
2. Click to expand the access token validator you want to edit.
3. Click the **Pencil** icon.
4. Make the desired edits. Click **Save**.

Deleting access token validators

Delete an existing access token validator in PingAccess.

Steps

1. Click **Access**, then go to **Token Validation > Access Token Validators**.
2. Click to expand the access token validator you want to delete.
3. Click the **Delete** icon.
4. To confirm your selection, click **Delete**.

Configuring OAuth key management settings

Configure settings for OAuth key management in PingAccess.

Steps

1. Click **Access**, then go to **Token Validation > OAuth Key Management**.
2. Choose to enable or disable key rolling:

   Choose from:

   - To enable key rolling, select the **Key Roll Enabled** check box.
   - To disable key rolling, clear the **Key Roll Enabled** check box.
3. To specify the interval at which you want to roll keys, enter a value (in hours) in the **Key Roll Enabled (H)** field.
4. From the **Signing Algorithm** list, select a signing algorithm to protect the integrity of the token when you use private key JSON Web Token (JWT) OAuth client authentication.

   If you select **Automatic**, you will use the algorithm specified in the OpenID Provider (OP) metadata.

5. Click **Save**.

Unknown resources

Unknown resources in PingAccess are resources that are not associated with an 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, you can configure agents to allow unprotected access instead of returning an error response.

Configuring unknown resource management

Define the action to take when an unknown resource is requested.

Steps

1. Click **Access**, then go to **Unknown Resources > Error Responses**.
2. In the **Error Status Code** field, enter the error status code for the HTTP response.

   This must be a client or server error code in the 400 - 599 range.
3. In the **Error Template File** field, enter 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. In the **Error Content Type** field, enter the file type of the response.

   Choose from:

   - HTML
   - JSON
   - TEXT
   - XML
5. To audit unknown resource activity, select the **Audit** check box.
6. Click **Save**.

Configuring agent defaults

Configure agents to allow unprotected access to unknown resources instead of returning an error response.

Steps

1. Click **Access**, then go to **Unknown Resources > Agent Defaults**.
2. Under the **Agent Default** header, specify the **Mode** that determines whether an agent should deny requests for unknown resources and generate an error response or allow requests to pass-through unfiltered.

   Individual agents might override this default setting.

3. Specify the default agent resource **Cache TTL (s)** (in seconds) to be used for unknown resources if you enable pass-through mode.
4. Click **Save**.

Managing risk policies

Create risk policies defining how PingAccess should respond to PingOne Protect's risk evaluations.

About this task

Currently, you can only create risk policies for the risk evaluation service provided by PingOne Protect. For a more detailed explanation of this integration, see PingOne Protect integration.

A risk policy tells PingAccess what action to take in response to the risk evaluations it receives from PingOne Protect. Apply a risk policy to a specific web application or resource to set up continuous authorization on your web applications with PingOne Protect.

To create or manage risk policies through the PingAccess administrative console, see:

Adding a risk policy

Adding a risk policy
Before you begin

Make sure that:

- You have set up a PingOne connection in PingAccess.
- You have your PingOne credential easily accessible to copy and paste.

For more information, see Adding a PingOne connection.

About this task

To add a risk policy:

Steps

1. In the PingAccess administrative console, go to **Access → Risk Policies** and click **+Add Risk Policy**.
2. Complete the fields.

   For more information, see Risk policy field descriptions.

   > **Note**
   >
   > You can only configure a PingOne risk policy in PingOne Protect.
   > If you haven't enabled device profiling in a PingAccess risk policy configuration, then you shouldn't include **New Device** or other device-related PingOne predictor types in the associated PingOne risk policy.
   > Some of these device-related predictor types are included in the default PingOne risk policy. If you haven't enabled device profiling, make sure to remove the following predictor types from your configuration or adjust the weights or scores associated with them:
   > - Anonymous network detection
   > - Geovelocity anomaly
   > - IP reputation
   > - IP velocity
   > - New device
   > - User location anomaly
   >
   > For more information, see Risk policies in the PingOne Cloud Platform documentation.

3. Click **Save**.

Next steps

After you've created a PingAccess risk policy, you can assign it to a specific application or resource. For more information, see Application field descriptions or Adding application resources.

Editing a risk policy

Editing a risk policy
Steps

1. Go to **Access → Risk Policies**.
2. Click the **Expand** icon to view more details about the risk policy that you want to edit.
3. On the **Properties** tab, click the **Pencil** icon.
4. Make the required changes.

   For more information, see Risk policy field descriptions.
5. Click **Save**.

Deleting a risk policy

Deleting a risk policy
Steps

1. Go to **Access → Risk Policies**.
2. Click the **Expand** icon to view more details about the risk policy that you want to delete.
3. Click the **Delete** icon.
4. Click **Delete**.

Risk policy field descriptions

The following table describes the fields available for managing risk policies on the **Risk Policies** tab in PingAccess.

Field	Required	Description
Name	Yes	A unique name for the risk policy.
PingOne Connection	Yes	The PingOne connection you created in steps 2a - 2c of Adding a PingOne connection.

Field	Required	Description
PingOne Risk Policy ID	No	The ID of the PingOne risk policy that you want to use to perform risk evaluation. A null value tells PingOne Protect to use a default policy. **Note** You can only configure a PingOne risk policy in PingOne Protect. If you haven't enabled device profiling in a PingAccess risk policy configuration, then you shouldn't include **New Device** or other device-related PingOne predictor types in the associated PingOne risk policy. Some of these device-related predictor types are included in the default PingOne risk policy. If you haven't enabled device profiling, make sure to remove the following predictor types from your configuration or adjust the weights or scores associated with them: • Anonymous network detection • Geovelocity anomaly • IP reputation • IP velocity • New device • User location anomaly Learn more in Risk Policies in the PingOne Cloud Platform documentation.

Field	Required	Description
Risk Check Interval (MS)	No	The rate at which PingAccess requests an evaluation from PingOne Protect for the same end user. This field accepts values from zero to a full day. The default value is 20000 ms (20 seconds). **Tip** To have PingOne Protect perform an evaluation on every request that an end user makes, you can set this value to `0`. However, evaluating every request could slow down your environment's performance.
User ID Attribute	Yes	Tells PingOne Protect what kind of user attribute to define as an end user's user ID.

High Risk Policy Evaluator	Yes	A policy that tells PingAccess what action to take if the returned risk score from an end user's request is `HIGH`. In the **High Risk Policy Evaluator** list, select one of the following options: ***Allow*** The default value. Permits the end-user's request. ***Authentication Challenge Policy*** Directs the user to reauthenticate. If you select this option, you must select an **Authentication Challenge Policy** to use. Adjusting the **Authentication Validity Period (M)** is optional. ***Deny*** Rejects the end-user's request. If you select this option, you must select a **Rejection Handler** to use. ***Rule*** PingAccess evaluates a rule you specify to determine how to proceed. If you select this option, you must select a specific web **Rule** to use.

Field	Required	Description
		Important API policy is currently incompatible with this type of policy evaluator. You can find more information on web policy and API policy in Applying rules to applications and resources. The following PingAccess rule types are API-specific and thus currently unusable on a protected web application: • OAuth attribute rules • OAuth client rules • OAuth Groovy script rules • OAuth scope rules • Rate limiting rules when the **Policy Granularity** of the rule is set to OAuth client • OAuth token cache time to live rules ***Rule Set*** PingAccess evaluates a rule set you specify to determine how to proceed. If you select this option, you must select a **Rule Set** to use.
Medium Risk Policy Evaluator	Yes	A policy that tells PingAccess what action to take if the returned risk score from an end-user's request is `MEDIUM`. In the **Medium Risk Policy Evaluator** list, select one of the five options described in the **High Risk Policy Evaluator** table entry.
Low Risk Policy Evaluator	Yes	A policy that tells PingAccess what action to take if the returned risk score from an end user's request is `LOW`. In the **Low Risk Policy Evaluator** list, select one of the five options described in the **High Risk Policy Evaluator** table entry.

Field	Required	Description
Failed Risk Policy Evaluator	Yes	A policy that tells PingAccess what action to take if the returned risk score is an invalid value or if the risk evaluation service is unavailable. In the **Failed Risk Policy Evaluator** list, select one of the five options described in the **High Risk Policy Evaluator** table entry.

Device Profiling Method	Yes	Specify if and how you want to collect an end-user's device profile. The default value is `OFF`. **Note** Device profiling helps PingOne Protect detect bot-like behavior and trigger step-up authentication in PingAccess. **Important** Device profile collection adds the device profile to the user's browser as cookies, which are sent to PingAccess during subsequent requests. These cookies are usually 8192 bytes in size. Before enabling device profiling, you should increase the `pa.default.maxHttpHeaderSize` property in the `<PA_HOME>/conf/run.properties` file to ensure a smooth transition. In the **Device Profiling Method** list, select one of the following options: ***OFF*** Select `OFF` if you don't want to perform device profiling. ***Captured by PingAccess*** Select **Captured by PingAccess** to have PingAccess perform device profiling. When this option is selected, PingAccess periodically interrupts end-user requests to display the **Device Profile Page**, an HTML page containing a script that collects the end-user's device profile.

Important
If you select **Captured by PingAccess**, the web page that you use to capture the device profile must be a `GET` request without a request body. The `GET` request must have an `Accept` header that allows text and HTML responses. You can also use the `pingone.protect.template.title` and `pingone.protect.template.header` properties in the `<PA_HOME>/conf/localization/pa-messages.properties` file to add messages. Learn more in User-facing page localization reference.

If you select **Captured by PingAccess**, the following fields become available:

- **Device Profile Interval**
- **Device Profile Timeout**
- **Device Profile Cookie Prefix**
- **Send Device Profile**

Captured by Frontend Application

Select **Captured by Frontend Application** to perform device profiling yourself without interrupting end-user requests. When this option is selected, you must set embed the PingOne Protect Signals SDK into your own web pages and send the device profile data that you collect to PingAccess using cookies. Learn more in the **Device Profile Cookie Prefix** table entry.

Field	Required	Description
		If you select **Captured by Frontend Application**, the following fields become available: • **Device Profile Cookie Prefix** • **Send Device Profile**
Device Profile Interval (S) **Note** This field is only available if you set **Captured by PingAccess** as the **Device Profiling Method**.	No	Define, in seconds, how frequently PingAccess should interrupt end-user requests to gather device profile data when the **Device Profiling Method** is set to **Captured by PingAccess**. This parameter accepts an integer value between 1 - 86400 seconds. The default value is `300` seconds.
Device Profile Timeout (MS) **Note** This field is only available if you set **Captured by PingAccess** as the **Device Profiling Method**.	No	Define, in milliseconds, how long the device profiling collection script will attempt to collect an end-user's device profile when the **Device Profiling Method** is set to **Captured by PingAccess**. If this timeout is exceeded, the script can't send device profile cookies to PingAccess, so PingAccess will follow the **Invalid Profile Risk Policy**. The default value is `5000` ms (5 seconds). A minimum value of `1000` ms (1 second) is required.

Field	Required	Description
Device Profile Cookie Prefix **Note** This field is only available if you set **Captured by PingAccess** or **Captured by Frontend Application** as the **Device Profiling Method**.	No	Define the cookie prefix used to send device profile data to PingAccess. The cookie prefix must be a valid token as described by RFC 6265. The default value is `p1_device_prof`. **Note** PingAccess expects sequential cookies using this cookie prefix and an index to provide the device profile data. For example, if you have three device profile cookies, you should order them in the following sequence: `p1_device_prof0=<first device profile segment>`, `p1_device_prof1=<second device profile segment>`, `p1_device_prof2=<third device profile segment>`. PingAccess concatenates these cookies and sends them to PingOne Protect when performing a risk evaluation for a user request.
Send Device Profile to Application **Note** This checkbox is only available if you set **Captured by PingAccess** or **Captured by Frontend Application** as the **Device Profiling Method**.	No	Select this checkbox if you want PingAccess to include device profile cookies in requests made to the protected application. This checkbox is cleared by default. **Note** Device profile cookies can be large and can sometimes make requests incompatible with backend servers.

Field	Required	Description
Sign Device Profile	No	Select this checkbox to sign and increase the security of device profiles captured by PingAccess and sent to PingOne Protect **Note** Signing device profiles can increase the header size of the data sent to PingOne Protect because of the signature size, and the serialization and JWT generation processes. Only applicable when the **Device Profiling Method** is **Captured by PingAccess**.
Invalid Profile Risk Policy	Yes	A policy that tells PingAccess what action to take in response to an end-user's request if the device profile information sent to PingAccess is invalid. For example, device profile information could be invalid because it's missing or because it isn't being collected as expected. In the **Invalid Profile Risk Policy Evaluator** list, select one of the five options described in the **High Risk Policy Evaluator** table entry.

Field	Required	Description
IP Change Enforcement	Yes	Specify the enforcement strategy that you want to use when PingAccess detects an IP address change from the end user. The default value is `NONE`. In the **IP Change Enforcement** list, select one of the following options: ***NONE*** PingAccess continues to allow user requests without recollecting device profile data or performing a new risk evaluation. ***Reevaluate Risk*** Performs a new PingOne Protect risk evaluation without recollecting the device profile. ***Collect Device Profile + Reevaluate Risk*** Collects the end user's device profile data again, then performs a new PingOne Protect risk evaluation.

To configure advanced settings, expand the **Show Advanced Settings** section at the bottom of the **Risk Policy** page. These settings are optional.

Field	Description
Device Profile Page **Note** You must set the **Device Profiling Method** to **Captured by PingAccess** to use this configuration option.	Specify the HTML template PingAccess should render if the **Device Profiling Method** is set to **Captured by PingAccess**. If you leave this field blank, PingAccess populates it with the `<PA_HOME>/conf/template/system/pingone.protect.template.html` default HTML template file after you save the risk policy. **Important** This default template contains the code that PingAccess uses to collect device profile data. Making changes to this template might interfere with PingAccess's ability to collect device profile data. You can make style changes to this template, but you should avoid making functional changes to it.

Field	Description
Max Expected Device Profile Cookies **Note** You must set the **Device Profiling Method** to **Captured by PingAccess** to use this configuration option.	Define the number of device profile cookies PingAccess should attempt to reset when it displays the **Device Profile** page. The default value is `5`. You must specify a value between 1 - 64. If PingAccess has seen the user before, it checks the user session data to determine the last set of device profile cookies it was sent and resets those cookies when it displays the device profile page. **Max Expected Device Profile Cookies** is only used when PingAccess is unable to determine the last set of device profile cookies that it was sent from the user. If you use the default **Device Profile Cookie Prefix**, `p1_device_prof`, PingAccess resets the cookies for `p1_device_prof0`, `p1_device_prof1`, `p1_device_prof2`, `p1_device_prof3`, and `p1_device_prof4` so the **Device Profile** page can edit them with the correct data.
Max Device Profile Retries	Define the maximum number of retry attempts to perform if PingOne Protect device profiling doesn't provide a response. For example, a retry attempt might be necessary because of browser incompatibilities. The default value is `5`.

Security header

The **Security** header contains controls for certificates and key pairs.

The **Security** header contains these menu options:

- Certificates
- Key pairs
- Hardware security module providers

Certificates

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 can be issued by a certificate authority (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 certificates

Import a new certificate.

Steps

1. Click **Security**, then go to **Certificates > Certificates**.
2. Click **+ Add Certificate**.
3. In the **Name** field, enter a name for the certificate.
4. To select the certificate, click **Choose File**.
5. To import the certificate, click **Add**.

> **Note**
>
> If the certificate is either expired or not yet valid, PingAccess displays a warning, but the import will proceed.

Result:

A new certificate row appears on the **Certificates** window.

Deleting certificates

Delete an existing certificate.

Steps

1. Click **Security**, then go to **Certificates > Certificates**.
2. Expand the certificate you want to delete.
3. Click the **Delete** icon.

4. When prompted, click **Delete** to confirm the deletion request.

> **Note**
>
> If the certificate is associated with a trusted certificate group, you cannot delete it.

Creating trusted certificate groups

Create a new trusted certificate group.

Steps

1. Click **Security**, then go to **Certificates > Trusted Certificate Groups**.
2. Click **+ Add Trusted Certificate Group**.
3. Drag a certificate into the box that appears.
4. In the **Name** field, enter a name for the group.
5. To set the new group to include the Java Trust Store group, select the **Use Java Trust Store** check box.

   Select this option if you create your own intermediate certificate authority (CA) certificate that is signed by a well-known CA in the Java Trust Store.
6. To allow PingAccess to ignore date-related errors for certificates that are not yet valid or have expired, select the **Skip certificate date check** check box.
7. To check the client certificate revocation status using certificate revocation list (CRL), select the **CRL checking** check box.
8. To check the client certificate revocation status using Online Certificate Status Protocol (OCSP), select the **OCSP** check box.

> **Note**
>
> If both certificate revocation list (CRL) checking and Online Certificate Status Protocol (OCSP) are enabled, OCSP checking is used preferentially, and CRL checking is used if OCSP fails.

9. To deny access when any certificate in the certificate chain cannot be verified using its CRL endpoint, select the **Deny when unable to determine revocation status** check box.
10. To validate client certificate chains that are not in the standard order, such as a reversed certificate chain of `[root, intermediate, leaf]`, select the **Validate disordered certificate chains** check box.
11. To skip validation of any CA certificates configured in the trusted certificate group and their subsequent chain of issuers when trusted CA certificates are found in the client certificate chain, select the **Bypass trust anchor validation** check box.
12. Click **Add**.
13. **Optional:** Add additional certificates to the new trusted certificate group by dragging them into the group.

> **Note**
>
> PingAccess has increased `WARN` logging during the certificate revocation check. You can adjust the log level using the AsyncLogger in `log4j2.xml` (search for "Certificate Revocation").
> A commented out `JAVA_SECURITY_OPTS` line is shipped as part of the `run.sh` and `run.bat` scripts. Uncommenting the `JAVA_SECURITY_OPTS` line enables extra java security logging/debugging for the PKIX CertPathValidator and CertPathBuilder implementations. You can use the `ocsp` option with the `certpath` option for OCSP protocol tracing.

Adding certificates to trusted certificate groups

Add a certificate to an existing trusted certificate group.

Steps

1. Click **Security**, then go to **Certificates > Trusted Certificate Groups**.
2. Drag a certificate into an existing trusted certificate group.

Editing trusted certificate groups

Edit the properties of an existing trusted certificate group.

Steps

1. Click **Security**, then go to **Certificates > Trusted Certificate Groups**.
2. Expand the trusted certificate group you want to edit.

Choice	Action
Add a certificate to the group	Drag it into the group from the certificate list.
Delete a certificate from the group	Click **–** to the right of the certificate.
Edit the trusted certificate group parameters	Click the **Edit** icon and then make your changes. If you edit these options, click **Save** to save them.

Removing certificates from trusted certificate groups

Remove a certificate from a trusted certificate group.

Steps

1. Click **Security**, then go to **Certificates > Trusted Certificate Groups**.
2. Click to expand the trusted certificate group containing the certificate you want to remove.
3. Click the **–** icon next to the certificate you want to remove.

Deleting trusted certificate groups

Delete an existing trusted certificate group.

Steps

1. Click **Security**, then go to **Certificates > Trusted Certificate Groups**.
2. Click to expand the trusted certificate group you want to delete.
3. Click the **Delete** icon.
4. To confirm the deletion request, click **Delete**.

Key pairs

PingAccess provides an interface for creating and managing 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 metadata about the owner of the private key.

The PingAccess administrative console displays a list of your configured key pairs. You can search for specific key pairs using the **Search** bar or filter your results using the **Filters** list.

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.

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 trust this certificate for authentication to succeed.

> **Note**
>
> 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. For more information, see the Oracle Java documentation.

Managing key pairs

Generate a key pair and self-signed certificate, import a key pair from a PKCS#12 or PEM-encoded file, or delete a configured key pair.

About this task

PEM-encoded key pair files use the following format for the key and certificates:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
 <Base64-encoded private key>
(Private Key:  <domain_name.key>)
-----END ENCRYPTED PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
 <Base64-encoded certificate>
(Primary SSL certificate:  <domain_name.crt>)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
 <Base64-encoded certificate>
(Intermediate certificate:  <Intermediate.crt>)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
 <Base64-encoded certificate>
(Root certificate:  <Root.crt>)
-----END CERTIFICATE-----
```

Importing existing key pairs

Importing existing key pairs
About this task

> **Note**
>
> If PingAccess is running in Federal Information Processing Standards (FIPS) mode, you can only import or export PEM-encoded key pairs. For more information, see Managing Federal Information Processing Standards (FIPS) mode.

To import a key pair from a PKCS#12 or PEM-encoded file:

Steps

1. Click **Security**, then go to **Key Pairs > 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 you're assigning the key pair to various configurations, such as HTTPS Listeners.
4. In the **Password** field, enter a password to protect the key pair file.

   PingAccess uses the password to read the file.
5. Click **Choose File** to locate the key pair 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. If the key pair cannot be read using the specified password, the import fails.

Generating new key pairs

Generating new key pairs
About this task

To generate a key pair and self-signed certificate:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click **+ Add Key Pair**.
3. In the **Alias** field, enter an internal alias for the key pair.
4. In the **Common Name** field, enter the common name identifying the certificate.
5. **Optional:** 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.
6. In the **Organization** field, enter the organization or company name of the group creating the certificate.
7. **Optional:** In the **Organization Unit** field, enter the unit within the organization.
8. **Optional:** In the **City** field, enter the city or primary location where the organization operates.
9. **Optional:** In the **State** field, enter the state or political unit where the organization operates.
10. In the **Country** field, enter the country where the organization operates.
11. In the **Valid Days** field, enter the number of days that the certificate is valid.
12. **Optional:** In the **Selected HSM** list, select a hardware security module to store the key pair in.
13. In the **Key Algorithm** section, select an algorithm:
    1. In the **Key Size** list, select the number of bits in the key.
    2. In the **Signature Algorithm** list, select the signature algorithm to use for the key.
14. Click **Save**.

Deleting key pairs

About this task

> **Note**
> If a key pair is currently in use, you cannot delete it.

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click to expand the key pair that you want to delete.

3. Click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Assigning key pairs

Assign a key pair to a virtual host or HTTPS listener.

About this task

PingAccess listens for HTTPS requests on the Admin, Engine, and Agent ports in all deployments, and on the Config query port in clustered deployments. See the Clustering in PingAccess reference guide for a comprehensive overview of the steps necessary to set up a clustered environment.

A key pair must be assigned to each listener. By default, the listeners are configured for HTTPS and use pregenerated 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 HTTP or HTTPS requests that are proxied to target web servers associated with Sites. For more information, see Engine listeners.
Agent	Listens for requests from PingAccess agents.
Sideband	Listens for requests from sideband clients.
Config query	Listens for requests for configuration information from replica administrative nodes and engine nodes in clustered deployments.

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.

Note

Cipher suite ordering for HTTPS listeners:

- PingAccess supports the use of a defined cipher suite order to ensure that the most secure cipher suites are used first, regardless of the client request. The cipher suite order is defined by the `tls.default.cipherSuites` property in the `<PA_HOME>/conf/run.properties` file.
- By default, new installations of PingAccess and environments upgraded to PingAccess 5.1 or later use this cipher suite ordering. To direct PingAccess to use the order provided by the client instead, use the PingAccess API `/httpsListeners` endpoint to set the `useServerCipherSuiteOrder` property to `false`.

To virtual hosts

Assigning key pairs to virtual hosts
About this task

To assign a key pair to a virtual host:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click the **Pencil** icon, and then click **Assign Virtual Host** for the key pair.
3. In the **Virtual Hosts** list, select the virtual hosts that you want to use the key pair with.

   > **Note**
   > When you assign a key pair to a virtual host, the key pair is also assigned to all other virtual hosts with the same host name.

4. Click **Save**.

To HTTPS listeners

Assigning key pairs to HTTPS listeners
About this task

To assign a new key pair for an active HTTPS listener:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click the **Pencil** icon, and then click **Assign HTTPS Listener** for the key pair.
3. In the **Listeners** list, select the HTTPS listeners that you want to use the key pair with.

   > **Note**
   > New connections use any of the changes you make to an HTTPS listener's active key pair, but existing connections continue to use the old configuration.

4. Click **Save**.

Managing key pair certificates

Add, download, or remove a certificate from a key pair, or manage key pairs using the automatic certificate management environment (ACME) protocol.

About this task

- Add a certificate to an existing key pair by starting with a leaf certificate and then adding the intermediate and root certificates as required.
- Remove a certificate from a configured key pair.
- Manage key pairs using the ACME protocol, which automatically obtains and renews certificates indirectly signed by a well-known trust anchor.
- Download a certificate when you need to configure a peer to trust a certificate used by PingAccess.

Adding certificates to key pairs

About this task

> **Note**
>
> To modify the certificates included in a chain, remove the certificates from the key pair and add them again. Alternatively, delete the certificate and recreate it by importing a new certificate file and adding certificates to the key pair.

To add a certificate to an existing key pair:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click to expand an existing key pair.
3. In the **Key Pair Chain Certificate** list, select **Add Certificate**.
4. To browse for and select the certificate file, click **Choose File**.
5. Click **Add**.

Removing certificates from key pairs

About this task

> **Note**
>
> Certificates can only be removed in reverse order. This procedure removes the last certificate in the chain.

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click to expand an existing key pair.
3. To remove the last certificate in the chain, click the **Delete** icon.
4. To confirm your changes, click **Delete**.

Managing certificates for key pairs with ACME

About this task

The ACME protocol is an Internet Engineering Task Force (IETF) proposed standard protocol that automates the signing of TLS certificates by a certificate authority (CA).

By default, the ACME certificate management option in PingAccess uses the staging Let's Encrypt ACME CA.

> **Note**
>
> The Let's Encrypt staging server, which PingAccess uses by default, has more lenient rate limits but it doesn't generate functional certificates, to support its use for testing purposes. For more information about rate limits, see the Let's Encrypt documentation.
> After testing your environment, you must switch to a production server using the PingAccess administrative application programming interface (API).
>
> 1. Use a `GET` call to `/pa-admin-api/v3/acme/servers` to retrieve the ID of a production server.
> 2. Use a `PUT` call to `/pa-admin-api/v3/acme/servers/default` to set the production Let's Encrypt server as the default.
>
> To add more ACME servers, use a `POST` call to `/pa-admin-api/v3/acme/servers`. For more information about the administrative API endpoints, see Administrative API endpoints.

To manage certificates with ACME:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click the **Pencil** icon, and then click **Manage with ACME** for the key pair.

   Result:

   The ACME status changes to **Pending**. When the protocol has completed, the status changes to **Valid** if the protocol completed successfully.

Downloading certificates

About this task

Download the certificate for the key pair used by a mutual TLS site authenticator and configure the target site to trust the certificate.

To download a certificate:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Locate the row corresponding to the key pair, and then click the **Pencil** icon.
3. Click **Download Certificate**.

   Result:

Your browser downloads the certificate and saves it in your local file system.

Managing certificate signing requests

Generate a certificate signing request (CSR) or import a CSR response.

About this task

- Generate a CSR to establish more security and trust than using a self-signed certificate.
- Import a CSR response to replace the self-signed certificate in a key pair.

Generating certificate signing requests

About this task

To generate a CSR:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click the **Pencil** icon, then click **Generate CSR** for the certificate that you want to generate a CSR for.

   Result:

   PingAccess generates a CSR file, and your browser downloads 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.

Next steps

When you receive the CSR response, follow the instructions in Importing certificate signing request responses.

Importing certificate signing request responses

Before you begin

Before you import the CSR response, import the signing CA certificate into PingAccess and add it to a Trusted Certificate Group.

About this task

To import a CSR response:

Steps

1. Click **Security**, then go to **Key Pairs > Key Pairs**.
2. Click the **Pencil** icon, and then click **CSR Response** for the key pair that the CSR applies to.
3. To select the CSR response file, in the **CSR Response File** section, click **Choose File**.

4. **Optional:** To choose one or more chain certificate files associated with this key pair, in the **Chain Certificates** section, click **Choose Files**.
5. Click **Save**.

Configuring static signing keys

Configure static keys for use in private key JSON Web Token (JWT) OpenID Connect (OIDC) code flow instead of dynamically rotating keys to sign tokens as necessary.

Before you begin

- In your token provider configuration, make sure that you've set up an OAuth client.

  If you haven't set up an OAuth client and are using PingFederate as the token provider, see managing OAuth clients.
- In PingAccess, make sure that you've generated or imported a key pair and then assigned it to a virtual host or HTTPS listener.

About this task

Static and dynamically rotating keys are used to sign self-contained access tokens, ID tokens, and JWTs for client authentication and OIDC request objects.

> **Important**
>
> You must make changes in both PingAccess and the token provider to modify your signing key configuration. Make these changes as soon as possible to reduce potential disruptions.

Dynamically rotating keys (default)

PingAccess generates and rotates keys automatically for OAuth and OpenID Connect.

> **Note**
>
> PingAccess uses the **Signing Algorithm** configured on the **OAuth Key Management** page for dynamic key rotation unless you have configured the signing algorithm on your web session. A signing algorithm configured on a web session takes priority over one configured on the **OAuth Key Management** page.

Static keys

Manually configure and rotate keys for OAuth and OpenID Connect to gain more control over key rotation.

To configure static signing keys:

Steps

1. In PingAccess, go to **Security > Key Pairs > Static OAuth/OIDC Keys**.
2. Select the **Enable Static Keys** check box to use static keys for OAuth and OpenID Connect.

   This check box is cleared by default.
3. In the **Signing Keys** section, fill out the relevant information for your static key configuration.

The **Active** and **Previous** lists only display signing keys that you've configured on the **Key Pairs** page that match the listed key type.

1. For the **RSA using SHA-256** key type, select a signing key in the **Active** list.

   > **Note**
   >
   > There are no default selections for the signing key lists. If you don't find the signing key that you want, go to the **Key Pairs** page and generate or import the desired type of key pair.

2. **Optional:** In the **Previous** list, select a signing key that you'd previously selected in the **Active** list if you still want the token provider to validate it.

   > **Note**
   >
   > If you select a certificate in the **Previous** list, that certificate will appear in the JWT, but only the **Active** certificate is actually used in a JWT signing flow.

3. **Optional:** Repeat steps 3a and 3b for each additional key type that you want to use.

4. **Optional:** For any key type for which you have selected an **Active** signing key, select the **Publish Certificate** check box to publish the certificates associated with the active signing key and the previous signing key (if configured) at the `GET /staticKeys/JWKS` endpoint.

   When you select the **Publish Certificate** check box for a key type, the associated chain of certificates is published as the `x5c` parameter value. This enables the OIDC provider to validate a certificate if it's been revoked.

5. Click **Save**.

Result:

The active signing key and the previous signing key (if configured) are published at the PingAccess static key JSON Web Key Set (JWKS) endpoint, `GET /staticKeys/JWKS`.

4. Prepare the token provider to validate the signed JWT that it will receive from PingAccess.

   > **Important**
   >
   > Switching between dynamically rotating and static keys in PingAccess doesn't work the same way as it does in PingFederate. If you change a dynamically rotating key to a static key in PingAccess, you can't use the previous **JWKS URL** value generated for the dynamically rotating key. This is because static keys and dynamically rotating keys use different JWKS endpoints in PingAccess. These endpoints generate values that overwrite each other.

   1. In PingAccess, on the **Static OAuth & OpenID Connect Keys** page, click **View Metadata**, then click **Copy**.

      > **Tip**
      >
      > Click **View Metadata** at any time to check the JWKS information available at the `GET /staticKeys/JWKS` endpoint.

   2. In your token provider environment, open the OAuth client that you're using for static key signing and paste the metadata value that you copied in step 4a into your JWKS configuration.

Example:

If you're using PingFederate as the token provider:

1. In PingFederate, go to **Applications → OAuth Clients** and open the OAuth client that you're using for this configuration.
2. In the **JWKS** field, paste the metadata value that you copied in step 4a.

For more information, see Configuring OAuth Clients.

Next steps

Configure the **Signing Algorithm** on the associated web session. For more information, see step 8 of Creating web sessions.

Hardware security module providers

Configure PingAccess to use a hardware security module (HSM) to generate and store key pairs to be used in SSL/TLS cryptographic operations.

PingAccess uses the HSM agent to direct the creation of new key pairs on the HSM. When you create a key pair, you can specify that it should be created on the HSM.

For more information, see the following topics:

- Adding an AWS CloudHSM provider
- Adding a Safenet Luna provider
- Editing an HSM provider
- Deleting an HSM provider

Adding an AWS CloudHSM provider

To use hardware security module (HSM)-stored key pairs in PingAccess, add an Amazon Web Services (AWS) CloudHSM provider in the PingAccess administrative console.

Before you begin

PingAccess 7.3 and later no longer support AWS CloudHSM Client SDK 3.

- If you're upgrading the CloudHSM Client SDK from 3.x to 5.x, see Upgrading from Client SDK 3 to Client SDK 5 before trying to add a CloudHSM provider in the PingAccess administrative console.
- If you are creating a new installation of AWS CloudHSM Client SDK 5, see Setting up a new installation of AWS CloudHSM before trying to add a CloudHSM provider in the PingAccess administrative console.

Important

Follow these steps to set up Client SDK 5 and integrate it with PingAccess even if you're just upgrading the Client SDK from 3.x to 5.x. Client SDK 5 no longer uses a client daemon. This changes the steps necessary to set up an AWS CloudHSM provider because the client process doesn't run separately from PingAccess anymore.

About this task

To add an AWS CloudHSM provider in the PingAccess administrative console:

Steps

1. In PingAccess, go to **Security → HSM Providers**, and click **+ Add HSM Provider**.
2. In the **Name** field, enter a name for the HSM provider.
3. In the **Type** list, select **AWS CloudHSM Provider**.
4. In the **User** field, enter a username used to connect to the HSM provider.
5. In the **Password** field, enter a password used to connect to the HSM provider.
6. **Optional:** In the **Partition** field, enter the partition to use on the HSM provider.
7. Click **Save**.
8. Restart PingAccess.

Troubleshooting

PingAccess 7.3 and later contain a workaround to bypass the following known issues by default:

1. `RSASSA-PSS` signing algorithms fail with `Java8u261` or later. HSM vendors and core Java use different naming conventions for the `RSASSA-PSS` algorithm.
2. PingAccess Cloud HSM functionality works in FIPS mode but not in regular mode for `Java8u261` and later.

If you experience either of these known issues, you can edit the `additional.security.jdk.tls.disabledAlgorithms` property in the `run.properties` file to bypass them. For more information, see the following example:

```
additional.security.jdk.tls.disabledAlgorithms=RSASSA-PSS, TLSv1.3
```

Next steps

Begin creating and assigning key pairs. For more information on creating key pairs, see Generating new key pairs or Importing existing key pairs.

Setting up a new installation of AWS CloudHSM

Before you begin

- Configure your hardware security module. You must have a AWS CloudHSM cluster to complete step 3. For more information, see the Amazon documentation.
- Ensure that Java 8 or 11 is installed on the PingAccess server. For more information on how to set up a Java Runtime Environment (JRE), see Installing PingAccess on your system. Make sure that you use a non-Oracle version of Java (such as Corretto), and if you use JDK 11, make sure that you use 11.0.16 or later.
- PingAccess must be deployed on an operating system that AWS CloudHSM supports. See System requirements in the PingAccess documentation and Supported platforms for the client SDKs in the AWS CloudHSM documentation for a list of mutually supported operating systems.

About this task

To set up a new installation of AWS CloudHSM Client SDK 5 and integrate it with PingAccess:

Steps

1. Request a crypto user (CU) account from your AWS CloudHSM administrator.

   > **Note**
   >
   > You will need to reference your username and password for this account during steps 4-5 of Adding an AWS CloudHSM provider. PingAccess uses this information to establish a connection with AWS CloudHSM.

2. Install and configure the AWS CloudHSM Java Cryptography Extension (JCE) provider for Client SDK 5.

   For more information, see Install and use the AWS CloudHSM JCE provider for Client SDK 5 in the AWS CloudHSM documentation.

   > **Important**
   >
   > You can't install the JCE provider if you already have the AWS CloudHSM client installed because of the structural changes made to the client between 3.x and 5.x. If you are upgrading from PingAccess 7.2 or earlier, you must remove any existing CloudHSM client software.

3. Connect the Client SDK to the AWS CloudHSM cluster.

   For more information on how to connect the Client SDK, see Bootstrap the Client SDK in the AWS CloudHSM documentation. Use the **JCE provider** tab.

4. Run the appropriate command for your operating system to ensure that keys are available to use.

   > **Note**
   >
   > You must complete this step even if you don't plan to use a cluster containing multiple HSMs.

   Choose from:

   - On Linux operating systems, run the `sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check` command.
   - On Windows operating systems, run the `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe --disable-key-availability-check` command.

5. If you plan to use elliptic curve (EC) keys for decryption, run the appropriate command for your operating system.

   Choose from:

   - On Linux operating systems, run the `sudo /opt/cloudhsm/bin/configure-jce --enable-ecdh-without-kdf` command.
   - On Windows operating systems, run the `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe --enable-ecdh-without-kdf` command.

6. Configure a new PingAccess installation on the network interconnected to the HSM.

   For more information on how to install PingAccess, see Installing PingAccess on your system.

> **Note**
>
> To integrate an existing PingAccess installation with your HSM, skip this step and proceed to step 7 instead.

7. To enable the Java interface and PingAccess integration, copy the `cloudhsm-jce-5.x.0.jar` file to the `pingaccess/deploy` directory.

> **Note**
>
> - On Linux operating systems, the file location is `/opt/cloudhsm/java/cloudhsm-jce-5.x.0.jar`.
> - On Windows operating systems, the file location is `C:\Program Files\Amazon\CloudHSM\java\cloudhsm-jce-5.x.0.jar`.

Next steps

Return to Adding an AWS CloudHSM provider to finish setting up an AWS CloudHSM provider in the administrative console.

Upgrading from Client SDK 3 to Client SDK 5

About this task

Upgrading from Client SDK 3 to Client SDK 5 requires you to have a source version of PingAccess that you plan to upgrade to or past a target version of PingAccess 7.3 or later.

To upgrade the AWS CloudHSM Client SDK from 3.x to 5.x to integrate it with a target version of PingAccess 7.3 or later:

Steps

1. Ensure that the source version of PingAccess that you plan to upgrade to or past version 7.3 is running.

> **Important**
>
> Do not stop this source version of PingAccess until step 7.

2. Stop the CloudHSM 3 standalone client with the `sudo service cloudhsm-client stop` command.
3. Move or delete the `/opt/cloudhsm` file.

> **Important**
>
> You can't complete step 4 if you already have the AWS CloudHSM client installed because of the structural changes made to the client between 3.x and 5.x. If you are upgrading from PingAccess 7.2 or earlier, you must remove any existing CloudHSM client software.

4. Install the JCE 5 client.

   For more information, see Install and use the AWS CloudHSM JCE provider for Client SDK 5 in the AWS CloudHSM documentation.

5. Copy the `cloudhsm-5.x.0.jar` file into the `pingaccess/deploy` directory of the target version of PingAccess that you plan to upgrade to.
6. Run the PingAccess upgrade.

   For more information, see Upgrading PingAccess.

7. Stop the source version of PingAccess.

   For more information, see Stopping PingAccess.

Result

You have upgraded to your target version of PingAccess and integrated AWS CloudHSM Client SDK 5 with it.

Adding a Safenet Luna provider

Add a Safenet Luna provider to begin using hardware security module (HSM)-stored key pairs in PingAccess.

Before you begin

- Configure your hardware security module.
- Configure a Luna client on the PingAccess system. The PingAccess service must have full permissions over the client.
- Move the `/usr/safenet/lunaclient/lib/libCryptoki2_64.so` library on Linux systems, or the `\Program Files\SafeNet\LunaClient\win32\cryptoki.dll` library on Windows systems, to the `deploy` directory on the PingAccess system.

Steps

1. Click **Security**, then go to **HSM Providers**.
2. Click **+ Add HSM Provider**.
3. In the **Name** field, enter a name for the HSM provider.
4. From the **Type** list, select **Safenet Luna Provider**.
5. In the **Slot ID** field, enter the slot ID of the HSM slot to use.
6. In the **Library** field, enter the name of the library you copied from the Luna client to the `deploy` directory.
7. In the **Password** field, enter a password for connecting to the HSM provider.
8. Click **Save**.
9. Restart PingAccess.

Editing an HSM provider

Edit the properties of an existing hardware security module (HSM) provider.

Steps

1. Click **Security**, then go to **HSM Providers**.
2. Click to expand the HSM provider, and then click the **Pencil** icon.
3. Edit one or more properties.
4. Click **Save**.

Deleting an HSM provider

Delete an existing hardware security module (HSM) provider.

Steps

1. Click **Security**, then go to **HSM Providers**.
2. Click to expand the HSM provider, and then click the **Delete** icon.
3. Click **Delete**.

Settings header

The **Settings** header contains menu options related to internal settings and configuration for PingAccess, such as clustering, networking, and authentication.

The **Settings** header contains these menu options:

- Clustering
- HTTP requests
- Networking
- Admin authentication
- System

Clustering

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

> **Important**
>
> Availability and performance are often at opposite ends of the deployment spectrum. You might need to make some configuration tradeoffs that balance availability with performance to accommodate specific deployment goals.

To learn more about how to set up cluster components in the PingAccess administrative console, choose one of the following sections:

- For more information on how to set up and manage the administrative nodes in a clustered PingAccess deployment, see Administrative nodes.
- For more information on how to set up and manage the engine nodes in a clustered PingAccess deployment, see Engines.
- For more information on how to configure new engine nodes to auto-register themselves at initial startup, see Engine Registration.

> **Note**
>
> These topics contain instructions to set up specific pieces of a cluster. For more information about clustering and setting up an efficient clustered environment, you should review the PingAccess Clustering in PingAccess *Reference Guide*. This guide provides information about subjects such as cluster node statuses, node failure implications, and which files could affect cluster settings.

Engines

Configure and manage the engine nodes in a cluster in PingAccess.

- Configuring engine nodes
- Editing engine nodes
- Revoking access from an engine node
- Removing engine nodes

Configuring engine nodes

Configure an engine node as part of a cluster in PingAccess.

Before you begin

Make sure that you've configured an administrative node and a replica administrative node.

> **Note**
>
> For a comprehensive overview of the steps necessary to set up a clustered environment, see Configuring a PingAccess cluster in the Clustering in PingAccess *reference guide*.

Steps

1. Click **Settings**, then go to **Clustering > Engines**.
2. To configure a new engine, click **+ Add Engine**.
3. In the **Name** field, enter a name for the engine.

   Special characters and spaces are allowed.
4. **Optional:** In the **Description** field, enter a description of the engine.
5. If applicable, specify an **HTTP Proxy** for the engine.

   For more information about creating proxies, see Adding proxies.

   1. To create an HTTP proxy, click **+Create**.
6. If applicable, specify an **HTTPS Proxy** for the engine.

For more information about creating proxies, see Adding proxies.

1. To create an HTTPS proxy, click **+Create**.

7. Specify an **Engine Trusted Certificate** if a TLS-terminating network appliance, such as a load balancer, is placed between the engines and administrative node.

> **Note**
>
> Select the certificate that the network appliance uses. The certificate helps establish a secure HTTP connection with the administrative node.

8. To generate and download a public and private key pair into the `<enginename>_data.zip` file for the engine, click **Save & Download**.

   This file is prepended with the name you give the engine. Depending on your browser configuration, you might be prompted to save the file.

9. Copy the `.zip` file to the `<PA_HOME>` directory of the corresponding engine in the cluster and extract it.

   The engine uses these files to authenticate and communicate with the administrative console.

> **Tip**
>
> You can generate a new key for the engine at any time, just repeat steps 8-9.
> 1. Click **Save & Download**.
> 2. Extract the `<enginename>_data.zip` file within the engine's `<PA_HOME>` directory.
>
> When the engine node starts up and begins using the new configuration files, PingAccess deletes the old key.

10. On Linux systems running the PingAccess engine, run the `chmod 400 conf/pa.jwk` command on the `pa.jwk` file after you've extracted the `.zip` file.

    Result:

    The `pa.jwk` becomes read only, preventing it from being overwritten accidentally.

11. Start each engine.

Next steps

If you specified any proxies, enable the **Use Proxy** option for any sites, token providers, and third party services that require the use of a proxy. For more information, see Adding sites and the Token provider section.

Editing engine nodes

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

Steps

1. Click **Settings**, then go to **Clustering > Engines**.
2. Expand the node you want to edit.
3. Click the **Pencil** icon.

4. Edit the node **Name** or **Description**, as appropriate.
5. If a new public key is needed, click **Save & Download**. If not, click **Save**.

Revoking access from an engine node

Remove an engine node's access to the administrative node.

About this task

If an engine node is 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. Click **Settings**, then go to **Clustering > Engines**.
2. Expand the engine node that you want to remove from the cluster and click the **Pencil** icon to edit it.
3. In the **Public Keys** section, click **Delete** to revoke the engine node's access to the administrative node.

> **Note**
>
> You can use the **Save & Download** button to create a new key for the engine. For more information, see Configuring engine nodes.

4. Click **Save**.

Removing engine nodes

Remove an engine node from the cluster.

Steps

1. Click **Settings**, then go to **Clustering > Engines**.
2. To permanently remove all references to the node from the cluster, expand the engine node you want to delete and click the **Delete** icon.
3. In the confirmation window, click **Delete**.

Engine Registration

Generate and download a file to enable a new engine node to automatically register itself at initial startup in PingAccess.

- Configuring engine nodes using an auto-registration file

Configuring engine nodes using an auto-registration file

Download a JSON Web Token (JWT) to enable a new engine node to automatically register itself at startup in PingAccess.

Before you begin

Make sure that you've configured an administrative node and a replica administrative node.

> **Note**
>
> For a comprehensive overview of the steps necessary to set up a clustered environment, see Configuring a PingAccess cluster in the Clustering in PingAccess *reference guide*.

Steps

1. Click **Settings**, then go to **Clustering > Engine Registration**.
2. If applicable, specify an **HTTP Proxy** for the engine.

   For more information about creating proxies, see Adding proxies.

   1. To create an HTTP proxy, click **+Create**.
3. If applicable, specify an **HTTPS Proxy** for the engine.

   For more information about creating proxies, see Adding proxies.

   1. To create an HTTPS proxy, click **+Create**.
4. Specify an **Engine Trusted Certificate** if a TLS-terminating network appliance, such as a load balancer, is placed between the engines and administrative node.

   > **Note**
   >
   > Select the certificate that the network appliance uses. The certificate helps establish a secure HTTP connection with the administrative node.
5. In the **Token Duration** field, enter the number of seconds for which the generated JWT is valid.
6. Click **Download**.
7. Add the JWT file to the engine node:

   Choose from:

   - If the engine node is not a container, copy the JWT file to the `<PA_Home>/conf` directory.
   - If the engine node is a container, inject the JWT as an environment variable named `REGISTRATION_TOKEN`.
8. **Optional:** Modify the name of the engine node:

   Choose from:

   - If the engine node is not a container, open the `<PA_Home>/conf/engine-registration.properties` file and update the *<engine.admin.api.engineName>* value.
   - If the engine node is a container, inject the engine node name as an environment variable named `ENGINE_NAME`.
9. Start the engine node.

Next steps

If you specified any proxies, enable the **Use Proxy** option for any sites, token providers, and third-party services that require the use of a proxy. For more information, see Adding sites and the Token provider section.

Administrative nodes

Configure and manage the administrative nodes in a cluster in PingAccess.

- Configuring administrative nodes
- Configuring replica administrative nodes

Configuring administrative nodes

Configure one PingAccess node as the administrative node.

About this task

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

In this procedure, you can specify an HTTP or HTTPS proxy. A proxy configuration defined in a properties file, such as `bootstrap.properties` or `run.properties`, takes precedence over a proxy configuration defined in the PingAccess administrative console or application programming interface (API).

If you configure a proxy on a replica administrative node and need to fail over to that node, make sure that the administrative node has the same proxy configuration as the replica administrative node before you remove the `bootstrap.properties` file from the replica administrative node.

For a comprehensive overview of the steps necessary to set up a clustered environment, see Configuring a PingAccess cluster in the Clustering in PingAccess *reference guide*.

Steps

1. Click **Settings**, then go to **Clustering > Administrative Nodes**.
2. In the **Host** field, in the **Primary Administrative Node** section, enter the host and port for the administrative console.

   The default is `localhost:9000`.
3. If applicable, specify an **HTTP Proxy** for the engine.

   For more information about creating proxies, see Adding proxies.

   1. Click **+ Create** to create an HTTP proxy.
4. If applicable, specify an **HTTPS Proxy** for the engine.

   For more information about creating proxies, see Adding proxies.

   1. Click **+ Create** to create an HTTPS proxy.

5. **Optional:** Select the **Enable Stale Engine Timeout** check box to remove engine node entities that have not communicated with the administrative node within the timeout value.

   > **Note**
   >
   > This option removes the engine node entity from the administrative node but does not modify the engine node itself.

   The default timeout value is 60 minutes. You can update this value through the PingAccess administrative API.

6. Click **Save**.

Next steps

If you specified any proxies, enable the **Use Proxy** option for any sites, token providers, and third party services that require the use of a proxy. For more information, see Adding sites and the Token provider section.

Configuring replica administrative nodes

Configure one PingAccess node as a replica administrative node to provide an alternative if the administrative node fails.

About this task

The key pair that you create for the CONFIG QUERY listener must include both the administrative node and the replica administrative node. To make sure the replica administrative node is included, you can either use a wildcard certificate or define subject alternative names in the key pair that use the replica administrative node's DNS name. For more information, see step 2c in Configuring a PingAccess cluster.

> **Important**
>
> 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.

Steps

1. Click **Settings**, then go to **Clustering > Administrative Nodes**.
2. In the **Host** field, in the **Replica Administrative Node** section, enter the host and port for the replica administrative node.

   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.

   For more information about creating proxies, see Adding proxies.

   1. Click **+ Create** to create an HTTP proxy.

4. If applicable, specify an **HTTPS Proxy** for the engine.

   For more information about creating proxies, see Adding proxies.

   1. Click **+ Create** to create an HTTPS proxy.

5. Specify the **Replica Administrative Node Trusted Certificate** if a TLS-terminating network appliance, such as a load balancer, is placed between the engines and administrative node.

   > **Note**
   >
   > Select the certificate that the network appliance uses. The certificate helps establish a secure HTTP connection with the administrative 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 in this window.

7. Copy the downloaded file to the replica administrative node's `<PA_HOME>` directory and extract it.

8. If the replica administrative node is running on a Linux host, run 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`.

   This property is case-sensitive.

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`.

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.

When a reverse proxy sits between the client and the PingAccess server or agent, the additional proxy might be identified as the client. You can configure such proxies to inject additional headers to relay the originating client address.

Host Source and **Protocol Settings** allow PingAccess to determine the effective Uniform Resource Locator (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.

IP Source

Lets you specify an ordered list of header names to identify the source Internet Protocol (IP) address. By default, `X-Forwarded-For` is configured as a heading.

Host Source

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.

Protocol Source

Can be used to define the header that identifies the protocol used for the original request. The default value is `X-Forwarded-Proto`.

Configuring alternative IP source headers

Configure an ordered list of Header names to identify the source IP address.

Steps

1. Click **Settings**, then go to **HTTP Requests > IP Source**.
2. From the **Header Names** list, search for the header names you want to add..

   You can add a header by clicking **+ Header Names** or delete a header by clicking the **Delete** icon.
3. In the **List Value Location** field, select either:

   Choose from:

   - `First`
   - `Last`

   When a list of values is in the header, this step determines if the first value or the last value in the list should be used as the IP Source value. The default value is `Last`.
4. To determine if you should use the upstream IP address for rule evaluation, if none of the listed headers are present in the request, either:

   Choose from:

   - Select the **Fallback to Last Hop IP** check box.
   - Clear the **Fallback to Last Hop IP** check box.

   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.
5. Click **Save**.

Configuring alternative host source headers

Configure an ordered list of Header names to identify the host source name in PingAccess.

About this task

Host source settings are valid for proxy deployments only.

Steps

1. Click **Settings**, then go to **HTTP Requests > Host Source**.
2. To enter a header name to search for in the **Header Names** list, click **Header Names**.

   You can add a header by clicking **+ Header Names** or delete a header by clicking the **Delete** icon.
3. In the **List Value** field, to determine (when a list of values is in the header) if the first or last value in the list should be used as the **Host Source** value, select one of the following:

   Choose from:

   - `First`
   - `Last`

   The default value is `Last`.
4. Click **Save**.

Configuring alternative protocol source headers

Configure a Header name for a protocol source in PingAccess.

Steps

1. Click **Settings**, then go to **HTTP Requests > Protocol Source**.
2. In the **Header Name** field, enter a header name.
3. Click **Save**.

Networking

The **Networking** tab controls how PingAccess manages network requests and load balancing.

For information related to networking in PingAccess, choose from one of the following sections:

- Availability profiles
- Engine listeners
- Managing load balancing strategies
- Proxies

Availability profiles

Use availability profiles in a site configuration to define how PingAccess classifies a backend target server as failed. Sites require you to select an availability profile, even if only one target server is provided.

You can determine connection failure based on whether a backend target isn't responding, or 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, you might consider that server down even though the web service is responsive.

If you specify multiple targets in a site configuration but don't apply a load balancing strategy to the site, the availability profile will cause PingAccess to use the first listed target in the site configuration unless that target fails. Secondary targets are only used if the first target isn't available.

> **Note**
>
> When you change the availability profile for a site, the new availability profile configuration only applies to new connections. Existing connections continue to use the previous availability profile configuration. To apply the new availability profile configuration to all connections for a site, you must restart the PingAccess engines.

Currently, the only availability profile type is **On-Demand**. You might want to create different profiles for different sites based on your site needs for retry counts, retry delays, timeouts, or HTTP status codes.

Creating availability profiles

Create an availability profile in PingAccess to define when a backend server is considered failed and how to handle failover.

Steps

1. Click **Settings**, then go to **Networking > Availability Profiles**.
2. Click **+ Add Availability Profile**.
3. In the **Name** field, enter a unique descriptive name for the profile.
4. In the **Type** list, select **On-Demand**.
    1. In the **Connect Timeout (MS)** field, enter the number of milliseconds PingAccess should wait for a connection to be established to the target site.

       The default value is `10000`.
    2. In the **Pooled Connection Timeout (MS)** field, enter the number of milliseconds PingAccess should wait before timing out a pooled connection request made to the target site.

       The default value is `-1`, no timeout.
    3. In the **Read Timeout (MS)** field, enter the number of milliseconds PingAccess should wait before timing out the read of the response from a target site.

       The default value is `-1`, no timeout.
    4. In the **Max Retries** field, enter a number specifying how many times PingAccess should try establishing a connection to a target site before considering the target site failed.

       The default value is `2`.
    5. In the **Retry Delay (MS)** field, enter the number of milliseconds PingAccess should wait between retries.

       The default value is `250`.

6. In the **Failed Retry Timeout (S)** field, enter a number, in seconds, specifying how long PingAccess should wait before trying to establish a connection to a failed target site again.

   The default value is `60`.

   > **Note**
   >
   > To retry a failed target site immediately, enter `0`.

7. (Optional) In the **Failure HTTP Status Codes** section, enter any HTTP status codes that PingAccess should consider a target site failure.

   To enter multiple HTTP status codes, click **+ New Value** for each status code.

   > **Note**
   >
   > The order of this list isn't important.

5. Click **Save**.

Editing availability profiles

Edit the properties of existing availability profiles in PingAccess.

Steps

1. Click **Settings**, then go to **Networking > Availability Profiles**.
2. Click to expand the desired profile you want to edit.
3. Click the **Pencil** icon.
4. Make the desired changes to the profile.
5. Click **Save**.

Deleting availability profiles

Delete existing availability profiles in PingAccess.

Steps

1. Click **Settings**, then go to **Networking > Availability Profiles**.
2. Click to expand the profile you want to delete.
3. Click the **Delete** icon.
4. To confirm your selection, click **Delete**.

Engine listeners

PingAccess listens for engine traffic using the defined engine listeners.

The default listener port is 3000. For more information on what you can do with engine listeners, see:

- Defining engine listeners
- Editing engine listeners
- Deleting engine listeners

Defining engine listeners

Define a new engine listener in PingAccess.

Steps

1. Click **Settings**, then go to **Networking > Engine Listeners**.
2. Click **+ Add Engine Listener**.
3. In the **Name** field, enter a descriptive name for the listener.
4. In the **Port** field, enter the port the listener will open.

   > **Note**
   >
   > If you do not open the port in the system firewall, the listener will not be able to process any incoming requests.

5. If you want the port to listen for HTTP connections, clear the **Secure** option.

   > **Note**
   >
   > By default, engine listeners listen for HTTPS connections to protect sensitive data.

6. **Optional:** From the **Client Certificate Authentication** list, select a certificate authentication method.

   Assigning a certificate authentication method to an engine listener provides a mechanism to authenticate using client certificates during any request to the engine listener.

   You can select an existing trusted certificate group, or use one of the following options.

   Choose from:

   - **No Certificate Authentication** – Does not require certificate authentication.
   - **Java Trust Store** – Uses the Java Trust Store for certificate authentication.
   - **Trust Any –** Allows client authentication with any certificate including self-signed certificates.

     If you use the **Trust Any** method in production, you should log client certificates in the audit log.

7. Click **Save**.

Editing engine listeners

Modify and edit the properties of an existing engine listener in PingAccess.

Steps

1. Click **Settings**, then go to **Networking > Engine Listeners**.
2. In the **Engine Listeners** section, click to expand an existing engine listener you want to edit.
3. Click the **Pencil** icon.
4. Enter the desired changes. Click **Save**.

Deleting engine listeners

Delete an existing engine listener in PingAccess.

Steps

1. Click **Settings**, then go to **Networking > Engine Listeners**.
2. In the **Engine Listeners** section, click to expand an existing engine listener you want to delete.
3. Click the **Delete** icon.
4. To confirm your selection, click **Delete**.

Managing 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.

Available load balancing strategies:

Header-Based

The header-based strategy requires incoming requests to include a header that defines the target to select from the **Site** configuration. This load balancing strategy allows fall back if the requested target is unavailable or if the header is missing from the request.

Round-Robin

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. The load balancer will not select a target that is in a failed state.

> **Note**
>
> If you specify multiple target servers for a site but don't apply a load balancing strategy, PingAccess uses the first target server in the list until it fails. Secondary target servers are only used if the first target server is not available. PingAccess uses the **Failed Retry Timeout** value from the site's availability profile settings to determine when to mark the first target server as available again.

Configuring load balancing strategies

About this task

To create and configure a new load balancing strategy in PingAccess:

Steps

1. Click **Settings**, then go to **Networking > Load Balancing Strategies**.
2. Click **+ Add Load Balancing Strategy**.
3. In the **Name** field, enter a unique descriptive name for the strategy.
4. In the **Type** list, select one of the following strategy types:

   Choose from:

   - **Header-Based**
   - **Round Robin**
5. Configure the options for the selected load balancing strategy type.

   Choose from:

   - For a **Header-Based** load balancing strategy:
     1. In the **Header Name** field, enter the name of the header that contains the selected target host.
     2. To configure PingAccess to use the first available target defined for the site, click **Show Advanced** and select the **Fall Back to First Available Host** option if the target specified in the header is not available or if the header is not present in the request.

        > **Note**
        >
        > If this option isn't enabled, and the specified target isn't available or the request header isn't present, the client receives a `Service Unavailable` response.
   - For a **Round Robin** load balancing strategy:
     1. If browser sessions shouldn't be pinned to a persistent backend target, clear the **Sticky Session Enabled** check box.

        This check box is selected by default.
     2. If **Sticky Session Enabled** is selected, enter a cookie name to use in the **Cookie Name** field.

        The PingAccess engine uses this cookie to track the persistent backend targets for a session.

> **Note**
>
> When you define a web session, 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.

6. Click **Save**.

Editing load balancing strategies

About this task

To edit the properties of an existing load balancing strategy in PingAccess:

Steps

1. Click **Settings**, then go to **Networking > Load Balancing Strategies**.
2. Click the **Expand** icon to see more details about the load balancing strategy that you want to edit.
3. Click the **Pencil** icon.
4. Make any desired changes to the properties.
5. Click **Save**.

Deleting load balancing strategies

About this task

To delete an existing load balancing strategy in PingAccess:

Steps

1. Click **Settings**, then go to **Networking > Load Balancing Strategies**.
2. Click the **Expand** icon to see more details about the load balancing strategy that you want to delete.
3. Click the **Delete** icon.
4. Click **Delete**.

Proxies

The **Proxies** page lets you configure the forward proxy configuration used when PingAccess makes requests to sites or token providers.

For more information on what you can do with proxies, see:

- Adding proxies
- Editing proxies
- Deleting proxies

Adding proxies

Add a forward proxy configuration to be used when PingAccess makes requests to sites or token providers.

Steps

1. Click **Settings**, then go to **Networking > Proxies**.
2. Click **+ Add Proxy**.
3. In the **Name** field, enter a name for the proxy configuration.
4. **Optional:** In the **Description** field, enter a description.
5. In the **Host** field, enter the host name for the forward proxy.
6. In the **Port** field, enter the port number for the forward proxy.
7. If the forward proxy requires authentication, select the **Requires Authentication** check box.
8. If required, enter the **Username** for the forward proxy.
9. If required, enter the **Password** for the forward proxy.
10. Click **Save**.

Next steps

If you have a clustered environment, configure your Admin and Engine nodes to use the proxy. If you are using a standalone environment, configure your Admin node to use the proxy. See Configuring administrative nodes and Configuring engine nodes for more information.

Then, enable the Use Proxy option for any sites, token providers, and third party services that require the use of a proxy. See Adding sites and the Token provider section for more information.

Editing proxies

Edit the properties for an existing proxy configuration in PingAccess.

About this task

> **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. Click **Settings**, then go to **Networking > Proxies**.
2. Click to expand an existing proxy configuration you want to edit.
3. Click the **Pencil** icon.
4. Edit the proxy as needed.

5. To confirm your edits, click **Save**.

Deleting proxies

Delete an existing proxy in PingAccess.

Steps

1. Click **Settings**, then go to **Networking > Proxies**.
2. Click to expand an existing proxy configuration you want to delete.
3. Click the **Delete** icon.
4. To confirm your selection, click **Delete**.

Admin authentication

Admin authentication controls the PingAccess administrator authentication method for the user interface and the admin APIs.

For information related to admin authentication in PingAccess, see:

- Configuring basic authentication
- Changing the password for basic authentication
- Configuring API authentication
- Admin UI SSO authentication
- Configuring admin UI session properties
- Configuring an admin token provider

Configuring basic authentication

Configure basic authentication for the administrative user interface in PingAccess.

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 user name and password credentials. The PingAccess server response contains a `PA_UI` cookie, which is a signed JSON Web Token (JWT). Subsequent HTTP requests send this cookie for authentication rather than the less secure HTTP Authorization header.

Basic Authentication supports one user: Administrator. The Administrator user name cannot be changed. If you want to allow more than one user to access the admin UI, you should use single sign-on (SSO) authentication.

Steps

1. Click **Settings**, then go to **Admin Authentication > Basic**.
2. Click **Enable**.

3. Click **Save**.
4. Click **Settings**, then go to **Admin Authentication > UI Authentication**.
5. In the **Authentication Method** section, select **Basic Authentication**.
6. Click **Save**.

Changing the password for basic authentication

Change the password used for basic authentication in PingAccess.

Steps

1. Click **Settings**, then go to **Admin Authentication > Basic**.
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 API authentication

Configure authentication for the administrative application programming interface (API) in PingAccess.

About this task

Learn more about the PingAccess Administrative API in Administrative API Endpoints.

You can configure roles for Administrative API users. Each role grants access to specific features:

- The Administrator role has full read and write access to the Admin API, unless the Platform Administrator role is enabled. If the Platform Administrator role is enabled, then the Administrator only has read access to the Admin API endpoints under the following paths, but has both read and write access to all other endpoints:
  - `/auth`
  - `/users`
  - `/environment`
- The Platform Administrator role has full read and write access to the Admin API. This role can be used with the Administrator role to grant full access to most features without the possibility of accidental lockout, with only the Platform Administrator able to change authorization configurations.
- The Auditor role has read access to all Admin API endpoints except for the following:
  - `/config/\*`

- `/backup/`
- `/agent/*/config/`

To configure API authentication:

Steps

1. Click **Settings**, then go to **Admin Authentication > API Authentication**.
2. To enable API OAuth authentication, select **OAuth**.
3. Click the **Properties** tab.
4. (Optional) In the **Access Token Validator Type** list, select a local access token validator to use instead of remote token validation for Admin API authentication.

   If you select a local access token validator, the OAuth configuration doesn't require client credentials or a subject attribute name.

   If you need to create a new access token validator, you can find more information about the field configurations in Adding access token validators.
5. Enter the **Client ID** assigned to you when you created the OAuth client for validating OAuth access tokens.

   Learn more about configuring a client ID in PingFederate in Configuring OAuth clients.
6. (Optional) Select a **Client Credentials Type**, then enter the information for the selected credential type.

   Choose from:

   - **Secret**: Enter the **Client Secret** you were assigned when you created the PingAccess OAuth client in the token provider.
   - **Mutual TLS**: Select a configured **Key Pair** to use for Mutual TLS client authentication.
   - **Private Key JWT**: Select this option to use Private Key JSON Web Token (JWT). No additional information is required.
7. Enter the **Subject Attribute Name** that 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.
8. Select the **Scope** required to successfully access the API.

   Learn more about defining scopes in PingFederate in Configuring authorization server settings.

   > **Note**
   >
   > If the administrative token is validated by a local access validator, the administrative API OAuth doesn't enforce whether an administrative token contains a `scope` claim with a configurable value.
9. (Optional) If you want to configure different DPoP settings for API authentication than the global DPoP settings that you configured in the token provider or admin token provider:

   1. Select **Override DPoP Settings**.

2. In the **DPoP Type** list, select the level of OAuth 2.0 Demonstrating Proof of Possession (DPoP) support that you want to enable for access token validation:
    - **Off** (default): PingAccess doesn't accept DPoP-bound access tokens, only bearer tokens.
    - **Enabled**: PingAccess accepts both bearer tokens and DPoP-bound access tokens.
    - **Required**: PingAccess doesn't accept bearer tokens, only DPoP-bound access tokens.
3. To require each DPoP proof to contain a nonce value during validation that was provided by PingAccess when the access token was created, per RFC 9449 section 9, select **Require Nonce**.

    This check box is cleared by default.
4. In the **DPoP Proof Lifetime (SEC.)** field, enter the duration, in seconds, that a DPoP proof should be considered valid after it's issued.

    > **Important**
    >
    > As a security best practice, keep this value low and consistent with the DPoP implementation of your API client. The default value is 120 seconds.

10. If you want to enable role-based authorization:
    1. Click the **Roles** tab.
    2. To enable role-based authentication, select **Enable Roles**.
    3. In the **Administrator** section, click **Add Required Attribute** as many times as you need.

        > **Note**
        >
        > For a role to be granted, all configured attribute values must match.

    4. Select an **Attribute Name** and **Attribute Value** for each required attribute.

        > **Tip**
        >
        > If you are using PingFederate as a token provider, the attribute name is defined in PingFederate under **OAuth Settings → Access Token Mappings → *Your_Policy* → Attribute Contact** as an extension to the contract.
        > The value that you use depends on the configuration of the **Contract Fulfillment** tab for the policy. Copy-paste your attribute value to ensure accuracy.

        Example:

        The attribute named `Group` in your attribute contract can be mapped to an Lightweight Directory Access Protocol (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 should use `Group` as the **Attribute Name** and `cn=pingaccess-admins,o=myorg` as the **Attribute Value**.
    5. In the **Matching Strategy** list, select the context that you want PingAccess to evaluate requests with when looking for a match.

Case-Sensitive

To register as a match, the attribute value in the request must be written in the same case as the attribute value that you specify in step 7. By default, PingAccess uses this matching strategy.

Case-Insensitive

Case doesn't matter when looking for a match. Select this option for more flexibility if you might make changes to the attribute source that don't alter it semantically.

DN Matching

Normalizes both the attribute value that you specify in step 7 and any attribute value that PingAccess gathers at runtime from the user identity attributes as an X.500 distinguished name (DN). PingAccess then looks for a match between the distinguished names.

> **Tip**
>
> - If you select **DN Matching**, make sure to select an **Attribute Name** in step 5 that can contain a DN. The administrative console doesn't provide a warning if you select an invalid attribute type, but you can check your log files to confirm that you don't have any DN-related errors.
> - PingAccess validates the **Attribute Value** that you specify in step 7 to make sure that it's a valid X.500 DN that follows RFC-1779 or RFC-2253. Copy-paste the attribute value to ensure accuracy.
> - Relative DNs that have non-printable or non-UTF-8 string values, such as email and domain component (DC) relative DNs, are case-sensitive. Otherwise, relative DN values are case-insensitive.

6. **Optional:** To add platform administrators:
    1. Select the **Enable Platform Administrator Role** check box.
    2. Select an **Attribute Name**, **Matching Strategy**, and **Attribute Value** for each required attribute.
    3. To add a new attribute, click **Add Required Attribute**.
7. **Optional:** To add auditors:
    1. Select the **Enable Auditor Role** check box.
    2. Select an **Attribute Name**, **Matching Strategy**, and **Attribute Value** for each required attribute.
    3. To add a new attribute, click **Add Required Attribute**.

11. Click **Save**.

Result

You have activated API authentication.

Admin UI SSO authentication

Configure single sign-on (SSO) for the administrative user interface in PingAccess.

To enable SSO, you must complete several configuration steps within the OpenID Provider (OP) and PingAccess.

> **Note**
>
> If you're using PingFederate as the token provider, you can configure the administrative SSO option to require a specific authentication mechanism. Configure PingAccess authentication requirements to use the OpenID Connect (OIDC) token provider's Requested AuthN Context Selector.

Preparing to configure admin UI SSO authentication

About this task

Before you can configure admin UI SSO authentication, you must:

Steps

1. Configure the OIDC provider.

   Choose from:

   - Configure PingFederate runtime.
   - Configure PingOne.
   - Configure OpenID connect.

2. Import the OIDC token provider server certificate into a trusted certificate group and associate that trusted certificate group with the OIDC token provider runtime.

   For more information, see Importing certificates.

3. If you're using PingFederate as the OIDC token provider, set up a **profile** scope in PingFederate that includes the openid, profile, address, email, and phone scope values.

   For more information, see Configuring OAuth clients in the PingFederate documentation.

   1. When you configure the client in PingFederate:

      - The **Client Authentication** must be set to anything except `None`.
      - The **Allowed Grant Types** must be set to `Authorization Code`.
      - The **Redirect URIs** must include `https://<PA_Admin_Host>:<PA_Admin_Port>/<reserved application context root>/oidc/cb`. The default reserved application context root is `/pa`.
      - If you're not using administrative roles in PingAccess, the OIDC **Policy** should be set to a policy that uses issuance criteria to restrict access based on some additional criteria.

        > **Warning**
        >
        > If the OIDC policy you select doesn't use issuance criteria to limit which users can be granted an access token, all users in the associated identity store configured in PingFederate can authenticate to the PingAccess administrative console and make changes.
        > For more information, see Defining issuance criteria for policy mapping in the *PingFederate Administrator's Manual*.

2. If you plan to use **Mutual TLS**, you must make two changes to the PingFederate configuration:
    1. Enable the use of the secondary HTTPS port in PingFederate by editing the `<pf_install>/pingfederate/bin/run.properties` file and setting the `pf.secondary.https.port` value to a port value. For more information, see Configuring PingFederate properties.
    2. Modify the `openid-configuration.template.json` file to add the `mtls_endpoint_aliases` object, with content defined by RFC-8705. For more information about this file, see the PingFederate documentation.

Configuring admin UI SSO authentication

Before you begin

If you're using PingFederate as the token provider, complete the configuration for connecting to the PingFederate OAuth authorization server (OAuth AS) on the Configuring PingFederate for PingAccess SSO page.

About this task

You can configure roles for PingAccess administrative console users. Each role grants access to specific features:

The Administrator role

Has full access to the UI unless the Platform Administrator role is enabled. If the Platform Administrator role is enabled, the Administrator can't update authorization, user, or environment settings, but can use all other features.

The Platform Administrator role

Has full access to all features. This role can be used with the Administrator role to grant full access to most features without the possibility of accidental lockout with only the Platform Administrator able to change authorization configurations.

The Auditor role

Can view the user interface but can't change the configuration.

To configure admin UI SSO:

Steps

1. Click **Settings**, then go to **Admin Authentication > UI Authentication**.
2. On the **Authentication Method** page, click **Single Sign-On**.

> **Tip**
>
> To define a fallback administrator authentication method if the OIDC token provider is unreachable, enable the `admin.auth=native` property in the `run.properties` file. This overrides any configured administrative authentication to basic authentication.

3. In the **OpenID Connect Login Type** list, select a sign-on type.

    Choose from:

    - **Code** (default): The standard OIDC sign-on flow.

- **POST**: A sign-on flow using the `form_post` response mode, which returns response parameters as `application/x-www-form-urlencoded` HTML form values.
- **x_post**: A sign-on flow based on OIDC that passes claims from the provider through 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. If you chose the **Code** sign-on type or want to enable session validation, select a **Client Credentials Type** from the following list, then provide the required information for the credential type that you chose.

   Choose from:

   - Click **Secret** to use a client secret. In the **Client Secret** field, enter the client secret assigned when you created the OAuth relying party client in the token provider.
   - Click **Mutual TLS** to use Mutual TLS client authentication. In the **Key Pair** list, select a configured key pair to use for Mutual TLS client authentication.
   - Click **Private Key JWT** to use Private Key JSON Web Token (JWT). No additional information is needed.

   > **Note**
   >
   > The OAuth client you use with PingAccess web sessions must have an OIDC policy specified. For more information, see Configuring OpenID Connect Policies.

6. If you want to include authentication requirements for your environment, then in the **Authentication Requirements** list, select a defined authentication requirements list or click **Create** to create a new list.
7. (Optional) In the **Username Attribute Name** field, enter the attribute from the ID token to be used as the display name in the user interface and included in the audit logs.

   If the attribute isn't specified or can't be found, the `sub` attribute is used.
8. (Optional) If you want to enable advanced settings, click **Show Advanced** and use one or more of the advanced options.

Advanced Option	Description
Scopes	Configure your token provider to handle all the requested scopes that you specify, including any custom scope values. ◦ To request one or more scopes from the OIDC token provider, in the **Scopes** list, select one or more scopes. **Note** If you configured a token provider, published scopes are available to select in the list based on the selected client ID. ◦ To specify unverified scopes, enter the scope and click **Use unverified scope "[scopename]"**. **Important** The user can access all attributes by examining browser traces. Although the attributes are integrity-protected to prevent changes, users can view any of your sensitive or confidential attributes if they can decode the ID token's value.
Validate Session	To validate sessions with the configured PingFederate instance during request processing, click **Yes**. **Note** This option is not supported by PingOne or third-party OIDC token providers.
Refresh User Attributes	To periodically refresh user data from the OIDC token provider: 1. Click **Yes**. 2. Specify a **Refresh User Attributes Interval** in seconds.
Cache User Attributes	To have PingAccess cache user attribute information for use in policy decisions, select the **Cache User Attributes** checkbox. **Note** When this option is disabled, user attribute information is encoded and stored in the session cookie.

Advanced Option	Description
Enable PKCE	To have PingAccess send a SHA256 code challenge and corresponding code verifier as a Proof Key for Code Exchange during the code authentication flow, select the **Enable PKCE** checkbox. **Note** The **OpenID Connect Login Type** must be set to **Code** for PingAccess to use PKCE.
Max Login Retries	Enter the maximum number of times PingAccess should retry a sign-on attempt after an authorization code exchange failure. To allow infinite retries, enter a value of `0`. To prevent PingAccess from retrying sign-on attempts, enter `-1`. The default value is `-1`. Applicable only when the **OpenID Connect Login Type** is **Code**.
Login Retry Delay (Sec.)	Enter the number of seconds PingAccess should wait before retrying a sign-on attempt after an authorization code exchange failure. The default value is `0`. Applicable only when the **OpenID Connect Login Type** is **Code** and a non-negative value is configured in the **Max Login Retries** field. Learn more in the previous step.
Use Single-Logout	To enable the use of single logout (SLO), select the **Use Single-Logout** checkbox. **Important** You must configure this option in the OIDC provider. **Tip** If you're using PingFederate as a token provider, enable the **Check For Valid Authentication Session** in the PingFederate access token management configuration to prevent session replay.

Advanced Option	Description
Include id_token_hint in SLO	Select the **Include id_token_hint in SLO** checkbox to include the `id_token_hint` parameter in the SLO request that PingAccess makes to the token provider. **Use Single-Logout** must be selected too. The `id_token_hint` parameter contains the PingAccess session cookie's `id_token`, which token providers can use to locate a user's session. Some token providers might require this parameter, but not all. **Important** If this option is enabled, PingAccess tracks the `id_token` attribute. Tracking the `id_token` attribute increases the PingAccess cookie's size. This could make the cookie exceed the browser's limit. For more information, see Minimizing the PingAccess cookie size. PingAccess can also track the `id_token` attribute if the **Track id_token** checkbox is enabled, but this isn't a prerequisite step to enabling **Include id_token_hint in SLO**. For more information on this configuration, see step 7c of Configuring OpenID Connect token providers.

9. (Optional) To enable role-based authorization:
    1. Click the **Roles** tab.
    2. To enable role-based authentication, select the **Enable Roles** checkbox.
    3. In the **Administrator** section, click **Add Required Attribute** for each attribute that you want to add.

       For a role to be granted, all configured attribute values must match.
    4. Select an **Attribute Name**, and **Attribute Value** for each required attribute.

       Note

       If you're 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 that you use depends on the configuration of the **Contract Fulfillment** tab for the policy. Copy-paste your attribute value to ensure accuracy.

       For example, the attribute named `Group` in your attribute contract can be mapped to an Lightweight Directory Access Protocol (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 should use `Group` as the **Attribute Name** and `cn=pingaccess-admins,o=myorg` as the **Attribute Value**.

5. In the **Matching Strategy** list, select the context that you want PingAccess to evaluate requests with when looking for a match.

   Case-Sensitive

   To register as a match, the attribute value in the request must be written in the same case as the attribute value that you specify in step 7. By default, PingAccess uses this matching strategy.

   Case-Insensitive

   Case doesn't matter when looking for a match. Select this option for more flexibility if you might make changes to the attribute source that don't alter it semantically.

   DN Matching

   Normalizes both the attribute value that you specify in step 7 and any attribute value that PingAccess gathers at runtime from the user identity attributes as an X.500 distinguished name (DN). PingAccess then looks for a match between the distinguished names.

   > **Tip**
   >
   > - If you select DN matching, make sure to select an **Attribute Name** in step 5 that can contain a DN. The administrative console doesn't provide a warning if you select an invalid attribute type, but you can check your log files to confirm that you don't have any DN-related errors.
   > - PingAccess validates the **Attribute Value** that you specify in step 7 to make sure that it's a valid X.500 DN that follows RFC 1779 or RFC 2253. Copy-paste the attribute value to ensure accuracy.
   > - Relative DNs that have non-printable or non-UTF-8 string values, such as email and domain component (DC) relative DNs, are case-sensitive. Otherwise, relative DN values are case-insensitive.

6. (Optional) To add platform administrators:
   1. Select the **Enable Platform Administrator Role** checkbox.
   2. Select an **Attribute Name**, **Matching Strategy**, and **Attribute Value** for each required attribute.
   3. To add a new attribute, click **Add Required Attribute**.
7. (Optional) To add auditors:
   1. Select the **Enable Auditor Role** checkbox.
   2. Select an **Attribute Name**, **Matching Strategy**, and **Attribute Value** for each required attribute.
   3. To add a new attribute, click **Add Required Attribute**.

10. Click **Save**.

Troubleshooting

You can find more information about what to do if you misconfigure admin UI SSO and get locked out in Administrative SSO lockout.

Configuring admin UI session properties

Configure session properties for the administrative console in PingAccess.

About this task

> **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**.

Steps

1. Click **Settings**, then go to **Admin Authentication > UI Session Properties**.
2. In the **Cookie Type** list, select a type of token to create.

   Choose from:

   - **Encrypted JWT** (default): An encrypted JSON Web Token (JWT) uses authenticated encryption to simultaneously provide confidentiality, integrity, and authenticity of the PingAccess token.
   - **Signed JWT**: A signed JWT uses asymmetric cryptography with a private-public key pairing to verify the signed message and confirm that it wasn't modified during transit.

   > **Note**
   >
   > Changing this setting could affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

3. In the **Audience** field, enter a short, unique identifier between 1 - 32 characters to define the audience to which the PingAccess token applies.

   The default value is `PingAccessUI`.

   Requests made to a target application that's associated with this web session must have a PingAccess token that matches the configured audience value. Otherwise, PingAccess redirects to the OIDC provider.

   > **Note**
   >
   > Changing this setting can affect existing ongoing sessions, forcing the user to reauthenticate to access protected resources.

4. In the **Idle Timeout** field, enter the amount of time, in minutes, that the PingAccess token remains active if PingAccess doesn't detect any user activity.

   The default value is `60` minutes. If an idle timeout occurs, PingAccess automatically terminates the associated session.

   > **Note**
   >
   > If the user has a valid existing PingFederate session when an idle timeout occurs in an associated PingAccess session, the PingAccess session might re-establish itself without prompting the user to sign on again.

5. In the **Max Timeout** field, enter a maximum amount of time, in minutes, that the PingAccess token remains active.

The default value is `240` minutes. When the PingAccess token expires, the associated user must reauthenticate. This protects against unauthorized resource use by ensuring that sessions end by the specified time and require the associated user to reauthenticate to continue.

> **Note**
>
> If PingFederate is the token provider, this value must be smaller than the PingFederate access token lifetime defined in the PingFederate access token management instance. Learn more in Reference token management.

6. In the **Expiration Warning** field, enter the amount of time, in minutes, before the session expires that PingAccess warns the user about the upcoming session expiration.

   The default value is `1` minute.

7. In the **Session Poll Interval** field, enter the amount of time, in seconds, that PingAccess waits between user info poll requests for the admin console.

   The default value is `10` seconds.

8. Select **Partitioned Cookie** to add the `Partitioned` attribute to the PingAccess admin console web session cookie.

   This ensures that cross-site cookies will continue to be readable within the same context that they're created in. Learn more in PingAccess 8.1 (June 2024).

   This checkbox is cleared by default.

   > **Note**
   >
   > Use the **Partitioned Cookie** checkbox to override the value of the `pa.default.session.cookie.attributes.partitioned` property in the `run.properties` file for the admin web session without needing to apply changes to all of the nodes in a PingAccess cluster and restart them.

9. Click **Save**.

Configuring an admin token provider

Configure a token provider to use when accessing the PingAccess user interface if you have enabled admin UI single sign-on or admin application programming interface (API) OAuth.

About this task

If you do not configure an admin token provider, the system token provider is used for both the PingAccess user interface and for end users.

Steps

1. Click **Settings**, then go to **Admin Authentication > Admin Token Provider**.
2. In the **Admin Token Provider** section, select **Admin**.
3. In the **Issuer** field, enter the issuer ID.
4. **Optional:** In the **Description** field, enter a description for the token provider.

5. In the **Trusted Certificate Group** list, select a trusted certificate group that PingAccess will use when authenticating to the admin token provider.
6. **Optional:** To configure the connection to use a configured proxy, click **Show Advanced Settings** and select **Use Proxy**.

   For more information about creating proxies, see Adding proxies.
7. To configure OAuth 2.0 Demonstrating Proof of Possession (DPoP) settings, click **Show Advanced Settings**:
   1. In the **DPoP Type** list, select the level of DPoP support that you want to enable for access token validation:
      - **Off** (default): PingAccess doesn't accept DPoP-bound access tokens, only bearer tokens.
      - **Enabled**: PingAccess accepts both bearer tokens and DPoP-bound access tokens.
      - **Required**: PingAccess doesn't accept bearer tokens, only DPoP-bound access tokens.
   2. To require each DPoP proof to contain a nonce value during validation that was provided by PingAccess when the access token was created, per RFC 9449 section 9, select **Require Nonce**.

      This check box is cleared by default.
   3. In the **DPoP Proof Lifetime (SEC.)** field, enter the duration, in seconds, that a DPoP proof should be considered valid after it's issued.

      > **Important**
      >
      > As a security best practice, keep this value low and consistent with the DPoP implementation of your API client. The default value is 120 seconds.
8. Click **Save**.

System

The **System** tab contains controls for the administrative UI in PingAccess.

To learn more about the actions available in the System tab, choose from one of the following sections:

- Configuration export/import
- License
- Token provider
- Environment
- System defaults
- Managing PingOne connections
- Log settings

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 is not older than the version used on the backup system, and as long as the backup and restore systems both use application programming interface (API) v3 (PingAccess 5.0 or later).

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 JSON web tokens (JWTs). It uses the same format as results from the administrative APIs.

> **Caution**
>
> Because the exported `JSON` file contains much of your PingAccess configuration, make sure that the file is safely 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 Add-on SDK for Java, the host key file (JWK set) is encrypted and included in the exported data. When running in Amazon Web Services (AWS), the AWS Key Management Services (KMS) master key encryptor bundled with PingAccess can be configured for master key encryption. See Configuring PingAccess to use Amazon Key Management Services for more information.

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 PingAccess configurations

Export your current configuration for PingAccess.

About this task

Large PingAccess configurations can take upwards of thirty minutes to export. During an export, you cannot modify the PingAccess configuration.

Steps

1. Click **Settings**, then go to **System > Configuration Export/Import**.
2. Click **Export Configuration**.

   The downloaded file name is `pa-data-<timestamp>.json`.

   > **Note**
   >
   > The *<timestamp>* value is formatted `MM-DD-YYYY.hh.mm.ss`. For example, a date and time of January 31, 2020 1:35 PM would be encoded as `01-31-2020.13.35.00` in the filename.

Importing PingAccess configurations

Import a previously saved configuration into your PingAccess instance.

About this task

The **Import Configuration** option is a version-specific tool that can import a previously exported configuration. PingAccess checks the exported `JSON` file to ensure that the file is not from a later version of PingAccess and is compatible with application programming interface (API) v3 (PingAccess 5.0 or later).

Large PingAccess configurations can take several hours to import. During an import, you cannot modify or read the PingAccess configuration.

> **Note**
>
> You can automatically import a configuration on a new system as part of the installation and startup process. For more information, see Installing PingAccess on Linux or Installing PingAccess on Windows from the command line.

Steps

1. Click **Settings**, then go to **System > Configuration Export/Import**.
2. Click **Import Configuration**.
3. Select the `JSON` export file that contains the configuration you want to import.
4. To start the import process, click **Import**.
5. When prompted for confirmation, click **Confirm**.

   > **Important**
   >
   > You might want to backup your system. 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.

   > **Note**
   >
   > If the import fails, click **View failures from last import** to view all of the errors logged during the import.

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.

License

View the details of your PingAccess license or upload a new license.

Go to **System → License** to display 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 the UI displays a warning ribbon on the page in certain cases, such as if the uploaded license will expire sooner than the current license. After reviewing any warnings, you can replace the existing license file by importing a new one.

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 administrator shuts down the server, removes the license file from the file system, and restarts the server, the existing runtime configuration will continue to work. However, the administrator will have to install a new license file on the file system, or upload one through the UI, to access and apply changes through the UI.

Uploading PingAccess licenses

Upload new license files in PingAccess.

About this task

When you select a new license file to import, PingAccess compares the new license file's attributes with the current license attributes before installing it. The UI displays a warning ribbon on the page in certain cases, such as if the uploaded license will expire sooner than the current license.

If the new license is acceptable, you can commit it and successfully replace the old license.

Steps

1. Click **Settings**, then go to **System > License**.
2. Click **Import License**.
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. To import the selected license, click **Import**.

   > **Note**
   >
   > If you select the wrong license, you can alter your selection either by clicking **Remove** to remove the selected license from the **Import License** section of the page, or by clicking **Choose File** to select a different license file.

5. To install the selected license, click **Confirm**.

Token provider

You can configure the token provider for PingAccess. The supported token providers are PingFederate, PingOne, PingOne Advanced Identity Cloud or PingAM, and common providers that use the OpenID Connect (OIDC) protocol.

Learn more about configuring PingFederate as the token provider for PingAccess in:

- Configuring a PingFederate runtime
- Configuring PingFederate administration
- Configuring OAuth resource servers
- Configuring PingFederate for PingAccess SSO

Learn more about configuring PingOne as the token provider for PingAccess in:

- Configuring PingOne

Learn more about configuring PingOne Advanced Identity Cloud or PingAM as the token provider for PingAccess in:

- Configuring PingOne Advanced Identity Cloud or PingAM as the token provider

Learn more about configuring a common token provider for PingAccess in:

- Configuring OpenID Connect token providers
- Configuring OAuth authorization servers

PingFederate

Configure an existing PingFederate environment as the token provider for PingAccess.

- Configuring a PingFederate runtime
- Configuring PingFederate administration
- Configuring OAuth resource servers
- Configuring PingFederate for PingAccess SSO

Configuring a PingFederate runtime

Configure an existing PingFederate environment as the token provider for PingAccess.

About this task

> **Note**
>
> For information on configuring PingFederate as an OAuth authorization server, see OAuth configuration and Configuring authorization server settings in the PingFederate documentation.

Before configuring a secure connection to the PingFederate runtime, export the PingFederate certificate and import it into a trusted certificate group in PingAccess:

Steps

1. In PingFederate, export the active certificate for the runtime server.

   For more information, see Manage SSL server certificates in the PingFederate documentation.

2. Import the certificate into PingAccess.
3. Create a trusted certificate group if one doesn't already exist.
4. Add the certificate to the trusted certificate group.

Next steps

Select the tab for your environment configuration to continue. If your PingFederate instance is proxied by the PingAccess engines, use the proxied runtime procedure. Otherwise, choose one of the standard runtime procedures.

The steps that display on the **Standard Runtime** tab in the PingAccess administrative console depend on what PingAccess version you're using:

- If you're using PingAccess 5.3 or later, some of the PingFederate configuration information is imported automatically from the PingFederate well-known endpoint. Use the standard runtime procedure.
- If you upgrade from PingAccess 5.2 or earlier and have an existing token provider configuration, you must provide the PingFederate configuration information manually. Use the original standard runtime procedure.

> **Tip**
>
> If you perform an upgrade from PingAccess 5.2 or earlier and want to see the updated version of the **Token Provider** page in the administrative console, configure the token provider using the `/pingfederate/runtime` application programming interface (API) endpoint. For more information, see Administrative API Endpoints.

> **Important**
>
> Configuring PingFederate as a token provider using the `/pingfederate/runtime` endpoint overwrites the existing PingFederate configuration.

Standard runtime

Configuring a standard PingFederate runtime

About this task

Configure a secure connection to the PingFederate runtime in PingAccess:

Steps

1. Click **Settings**, then go to **System > Token Provider > PingFederate > Runtime**.
2. Select **Standard Token Provider**.
3. In the **Issuer** field, enter the PingFederate issuer name.
4. **Optional:** In the **Descriptions** field, enter a description for the PingFederate instance.
5. In the **Trusted Certificate Group** list, select the certificate group that the PingFederate certificate is in.
6. To configure advanced settings, click **Show Advanced**.
    1. If host name verification for secure connections isn't required for either the runtime or the backchannel servers, select the **Skip Hostname Verification** check box.
    2. To use a configured proxy for backchannel requests, select the **Use Proxy** check box.

        > **Note**
        >
        > If the node is not configured with a proxy, requests are made directly to PingFederate. For more information about creating proxies, see Adding proxies.

    3. Select **Use Single-Logout** to enable single logout (SLO) when the `/pa/oidc/logout` endpoint is accessed to clear the cookie containing the PingAccess token.

        If you select this option, PingAccess sends a sign off request to PingFederate, which completes a full SLO flow.

        To use this feature, SLO must be configured on the OpenID Provider (OP).

    4. Enter the **STS Token Exchange Endpoint** to be used for token mediation if it's different from the default value of `<issuer>/pf/sts.wst`.
7. Click **Save**.

    > **Note**
    >
    > Saving a new PingFederate runtime configuration overwrites any existing PingFederate runtime configuration.

Result

After you save the PingFederate runtime connection, PingAccess tests the connection to PingFederate. If the connection can't be made, a warning displays in the admin console, and the PingFederate runtime won't save.

Next steps

After you save this configuration and perform the steps in Configuring OAuth resource servers, a PingFederate access validator is available for selection when you define OAuth-type rules in the policy manager.

After you configure the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **Refresh Metadata**.

Original standard runtime

Configuring a standard PingFederate runtime (original workflow)
About this task

If you've upgraded your PingAccess deployment from version 5.2 or earlier with an existing token provider configuration and haven't configured a token provider using the `/pingfederate/runtime` API endpoint, use this workflow to configure a PingFederate runtime.

Steps

1. Click **Settings**, then go to **System > Token Provider > PingFederate > Runtime**.
2. Select **Standard Token Provider**.
3. In the **Host** field, enter the PingFederate runtime host name or the IP address for the PingFederate runtime.
4. In the **Port** field, enter the PingFederate runtime port number.
5. **Optional:** In the **Base Path** field, enter the base path for the PingFederate runtime.

   The base path must start with a slash, such as `/federation`.
6. Select 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.
7. In the **Secure** section, select **Yes** if PingFederate is expecting HTTPS connections.
8. In the **Trusted Certificate Group** list, select the certificate group that the PingFederate certificate is in.

   > **Note**
   > This field is available only if you select **Yes** in step 7.
9. Click **Show Advanced** and configure the advanced settings:
   1. Click **Add Back Channel Server**.
   2. In the **Back Channel Servers** list, enter one or more `<hostname:port>` pairs.
   3. If the backchannel uses HTTPS, enable the **Back Channel Secure** option.

      This option is available after you define at least one backchannel server.
   4. If the backchannel uses an alternate base path, enter the path in the **Back Channel Base Path** field.
   5. If host name verification for secure connections isn't required for either the runtime or the backchannel servers, enable the **Skip Hostname Verification** option.
   6. If host name verification is required, enter the host name that PingAccess should expect in the **Expected Hostname** field.
   7. To use a configured proxy for backchannel requests, select the **Use Proxy** check box.

> **Note**
>
> If the node is not configured with a proxy, requests are made directly to PingFederate. For more information about creating proxies, see Adding proxies.

8. Select **Use Single-Logout** to enable single logout (SLO).

   To use this feature, SLO must be configured on the OpenID Connect (OIDC) provider.

10. Click **Save**.

Result

After you save the PingFederate runtime connection, PingAccess tests the connection to PingFederate. If the connection can't be made, a warning displays in the admin console, and the PingFederate runtime won't save.

Next steps

After you save this configuration and perform the steps in Configuring OAuth resource servers, a PingFederate access validator is available for selection when you define OAuth-type rules in the policy manager.

After you configure the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **Refresh Metadata**.

Proxied runtime

Configuring a proxied PingFederate runtime

About this task

Configure a secure connection to the proxied PingFederate runtime in PingAccess:

Steps

1. Click **Settings**, then go to **System > Token Provider > PingFederate > Runtime**.
2. Click **Proxied Token Provider (PingFederate Runtime Application)**.
3. In the **Primary Virtual Host** field, enter the virtual host to use for the PingFederate application.

   If you haven't created the virtual host, click **+ Create**. For more information, see Creating new virtual hosts.

   This virtual host is used by default for front-channel redirects to the PingFederate token provider when an application-specific OpenID Connect (OIDC) issuer isn't defined.
4. **Optional:** In the **Additional Virtual Hosts** field, enter one or more virtual hosts that can be used for the PingFederate application.

   If you haven't created the virtual host, click **+ Create**. For more information, see Creating new virtual hosts.
5. In the **Targets** field, enter a `hostname:port` pair used to access the PingFederate runtime servers.

   Click **+ Add Target** to add additional **Targets** fields.
6. In the **Secure** section, click **Yes** if the PingFederate runtime expects HTTPS connections.
7. In the **Trusted Certificate Group** list, select the certificate group the PingFederate certificate is in.

   > **Note**
   >
   > This field is available only if you select **Yes** in step 6.
8. In the **Availability Profile** list, select the availability profile that the PingFederate runtime should use.

   To create a new availability profile, click **+ Create**.
9. To record requests to PingFederate to the audit store, select the **Audit** check box.

   This check box is selected by default.
10. **Optional:** To configure advanced settings, click **Show Advanced**.

Option	Description
Context Root	Enter the first part of the URL path for the PingFederate application and its resources. The context root must begin with a slash. It can contain additional slashes, but cannot end with one. It must match the path defined by the base URL in PingFederate.

Option	Description
Case Sensitive	Select this check box to make the context root and resource path matching case sensitive.
Client Certificate Header Name	In this section, click **+ Add Client Certificate Header Name** and enter one or more header names to which PingAccess should map client certificates found in the request. The position of the header name in the list correlates to the index in the client certificate chain, with the first header mapped to the leaf certificate.
Policy	In this section, add one or more rules, rule sets, or rule set groups to run when making requests to the PingFederate runtime. ◦ Click **Rules**, **Rule Sets**, or **Rule Set Groups**, then drag one or more selections from the **Available** column to the **Selected Policy** column. Valid rule types are Groovy script, cross-origin request, and rewrite rules. ◦ Create new rules, rule sets, or rule set groups by clicking **+ Create Rule**, **+ Create Rule Set**, or **+ Create Rule Set Group**.
Load Balancing Strategy	In this list, select a load balancing strategy to use for requests to the PingFederate runtime. If you specify multiple target servers for a proxied PingFederate runtime but don't apply a load balancing strategy, PingAccessuses the first target server in the list until it fails. Secondary target servers are only used if the first target server is not available. PingAccess uses the **Failed Retry Timeout** from the runtime's availability profile settings to determine when to mark the first target server as available again.
Expected Certificate Hostname	Enter the host name expected in the certificate. If this field isn't specified, certificates are verified using the target host names.
Skip Hostname Verification	Click to stop the backchannel servers from performing host name verification of the certificate.
Use Proxy	Click to make backchannel requests to PingFederate use the proxy configured on the PingAccess nodes.

Option	Description
Use Single-Logout	Click to enable single logout if it's configured for the OP.

11. Click **Save**.

> **Note**
>
> Saving a new PingFederate runtime configuration overwrites any existing PingFederate runtime configuration.

Result

After you save the PingFederate runtime connection, PingAccess tests the connection to PingFederate. If the connection can't be made, a warning displays in the admin console, and the PingFederate runtime won't save.

Next steps

After you save this configuration and perform the steps in Configuring OAuth resource servers, a PingFederate access validator is available for selection when you define OAuth-type rules in the policy manager.

After you configure the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **Refresh Metadata**.

Configuring PingFederate administration

Configure your PingFederate administration settings in the PingAccess administrative console.

About this task

For information on the PingFederate administration application programming interface (API), see PingFederate administrative API.

When you save the PingFederate administration configuration, PingAccess will test the connection to PingFederate. If PingAccess can't make a connection, an error will display in the administrative console and the configuration won't save.

Steps

1. Click **Settings**, then go to **System > Token Provider > PingFederate > 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 runtime.
4. If necessary, enter the **Base Path** for the PingFederate runtime.

The **Base Path** must start with a slash (/).

Example:

`/path`.

5. If the PingFederate administrative API requires native authentication, click **Basic**.
    1. Enter the **Admin Username**.

       This username only requires auditor (read-only) permission in PingFederate.
    2. Enter the **Admin Password**.
6. If the PingFederate administrative API requires OAuth 2.0 authentication, click **OAuth**.
    1. In the **Configured Authorization Server** list, choose from:
        - **PingFederate Runtime**
        - **Admin Token Provider** (will only display if configured)

        > **Note**
        >
        > The API endpoint `/pingfederate/admin` allows you to select additional options for the configured authorization server.
        > You can configure the following authorization servers in the PingAccess administrative console:
        > - **PingFederate Runtime**. For more information, see Configuring a PingFederate runtime.
        > - **Admin token provider**. For more information, see Configuring an admin token provider.
        > - **Common**. For more information, see Configuring OAuth authorization servers.
        > - **PingOne**. For more information, see PingOne.

    2. In the **Client ID** field, enter a client ID for the OAuth client configured in the token provider.

       Choose a client that is configured with the client credentials grant type.
    3. In the **Client Credentials Type** field, select the credentials for the OAuth client configured in the token provider.
    4. In the **Scopes** field, enter the required scopes of validated access tokens that are authorized to call the PingFederate administrative API.

       > **Note**
       >
       > Scopes can be input as an array of case-sensitive strings. For a full list of the required scopes, see PingFederate's `required.scopes` section of the `oauth2.properties` file.
7. To log information about the transaction to the audit store, select **Audit**.

   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. In the **Secure** section of the **Administration** tab, click **Yes** if PingFederate is expecting HTTPS connections.

   Otherwise, click **No**.
9. From the **Trusted Certificate Group** list, select the group of certificates to use when authenticating to PingFederate.

PingAccess requires the certificate that PingFederate is using to anchor to a certificate in the associated trusted certificate group.

This field is available only if you enable **Secure** connections in step 8.

10. **Optional:** To configure advanced settings, click **Show Advanced**.
    1. Select **Skip Hostname Verification** to not perform hostname verification of the certificate.
    2. Enter an **Expected Certificate Hostname** to verify the certificate with the specified name instead of the **Host** name.
    3. To use a configured proxy for API requests, select the **Use Proxy** check box.

    > **Note**
    > If the node isn't configured with a proxy, requests are made directly to PingFederate.

11. Click **Save**.

> **Tip**
> To view OpenID Connect (OIDC) metadata provided by the token provider, click **View Metadata** after saving the token provider configuration.

Configuring OAuth resource servers

When receiving OAuth-protected application programming interface (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.

Before you begin

Before configuring an OAuth resource server, you must finish configuring the PingFederate administration.

If you plan to use **Mutual TLS**, you must make two changes to the PingFederate configuration:

1. Enable the use of the secondary HTTPS port in PingFederate by editing the `<pf_install>/pingfederate/bin/run.properties` file and setting the `pf.secondary.https.port` value to a port value. Learn more in Configuring PingFederate properties.
2. Modify the `openid-configuration.template.json` file to add the `mtls_endpoint_aliases` object, with content defined by RFC-8705. Learn more information about this file in the PingFederate documentation.

About this task

To validate the bearer access token, a valid OAuth client must exist within the PingFederate OAuth authorization server.

> **Note**
> This configuration is optional and necessary only if you plan to validate PingFederate OAuth access tokens.

To configure an OAuth resource server:

Steps

1. Click **Settings**, then go to **System > Token Provider > PingFederate > OAuth Resource Server**.
2. Enter the OAuth **Client ID** that you defined when creating the PingAccess OAuth client in PingFederate, as shown in Configuring OAuth clients.

   > **Note**
   >
   > Confirm that you selected **Access Token Validation** as the allowed grant type when configuring the OAuth client in PingFederate.

3. Select a **Client Credentials Type**, then provide the information required for the selected credential type.

   Choose from:

   - **Secret**: Enter the **Client Secret** assigned when you created the PingAccess OAuth client in PingFederate.
   - **Mutual TLS**: Select a configured **Key Pair** to use for Mutual TLS client authentication.
   - **Private Key JWT**: Select this option to use Private Key JSON web token (JWT). No additional information is required.

4. In the **Cache Tokens** section of the **OAuth Resource Server tab**, select whether to retain token details for subsequent requests.

   Choose from:

   - **Yes**: Selecting **Yes** reduces communication between PingAccess and PingFederate.
   - **No**: The default value.

5. If **Cache Tokens** is enabled, enter the number of seconds to cache the access token in the **Token Time To Live** field.

   The default value, `-1`, caches the token as long as it's valid.

6. In the **Subject Attribute Name** field, enter the attribute from the OAuth access token that you want to use as the subject for auditing purposes, such as `username`.

   At runtime, the attribute's value is used as the **Subject** field in the audit log entries for API Resources with policies that validate access tokens. The attribute you choose must align with an attribute in the OAuth access token attribute contract that's defined in PingFederate.

7. If multiple access token managers are configured in PingFederate, select the **Send Audience** check box to send the URI that the user requested as the `aud` OAuth parameter to select a specific access token manager.

   > **Note**
   >
   > This option requires you to configure the access token management instances with appropriate Resource URIs. Resource Uniform Resource Identifier (URI) matching is performed on a most-specific match basis.

8. **Optional:** To disable the use of OAuth 2.0 token introspection, clear the **Use Token Introspection Endpoint** check box.
9. In the **DPoP Type** list, select the level of OAuth 2.0 Demonstrating Proof of Possession (DPoP) support that you want to enable for access token validation:

Choose from:

- **Off** (default): PingAccess doesn't accept DPoP-bound access tokens, only bearer tokens.
- **Enabled**: PingAccess accepts both bearer tokens and DPoP-bound access tokens.
- **Required**: PingAccess doesn't accept bearer tokens, only DPoP-bound access tokens.

> **Note**
>
> You must use PingFederate 11.3 or later to configure DPoP support.

10. To require each DPoP proof to contain a nonce value during validation that was provided by PingAccess when the access token was created, per RFC 9449 section 9, select **Require Nonce**.

    This check box is cleared by default.

11. In the **DPoP Proof Lifetime (SEC.)** field, enter the duration, in seconds, that a DPoP proof should be considered valid after it's issued.

> **Important**
>
> As a security best practice, keep this value low and consistent with the DPoP implementation of your API client.

12. Click **Save**.

Configuring PingFederate for PingAccess SSO

Configure PingFederate to enable administrator single sign-on (SSO) for PingAccess.

Before you begin

You must do one of the following:

- Configure a PingFederate runtime.
- Configure an admin token provider.

About this task

To enable administrator SSO to PingAccess, configure the following settings within the PingFederate OAuth authorization server (OAuth AS).

> **Note**
>
> This document doesn't cover all the required steps for each PingFederate OAuth settings page, only the fields that are necessary for successful SSO to the PingAccess administrative console.
> For more detailed configuration information on the PingFederate OAuth settings pages, see Using OAuth Menu Selections.

Steps

1. In PingFederate, go to **System → Server → Protocol Settings → Roles and Protocols** and configure the following roles and protocols:
    1. Select the **OAuth 2.0 AS** federation role and the OpenID Connect (OIDC) protocol as described in step 2 of Choosing roles and protocols.
    2. Select the **IdP Provider** federation role and a corresponding protocol as described in step 2 of Choosing roles and protocols.
2. Create a **Password Credential Validator (PCV)** to authenticate administrative users.

    For more information, see Configuring the Simple Username Password Credential Validator.
3. On the **IdP Adapters** page, create an HTML Form IdP Adapter and specify the PCV that you configured in step 2 of this procedure.

    For more information, see Configuring an HTML Form Adapter instance.
4. On the **Authorization Server Settings** page, select the **Implicit** check box in the **Reuse Existing Persistent Access Grants for Grant Types** section.

    For more information, see Configuring authorization server settings.
5. Configure access token management:
    1. Go to **Access Token Management → Type** and in the **Type** list, select **Internally Managed Reference Tokens**.
    2. On the **Access Token Attribute Contract** page, add the `Username` attribute to extend the contract.

    For more information, see Access token management.
6. Configure **OpenID Connect Policy Management**.

    > **Note**
    > Create an OIDC policy to use specifically for PingAccess administrative console authentication.

    For more information, see Configuring OpenID Connect policies.
    1. On the **Attribute Contract** tab, delete all of the attributes that appear in the **Extend the Contract** section.

        The only required attribute is `sub`.
    2. On the **Contract Fulfillment** tab, in the **Source** list, select **Access Token**, and in the **Value** list, select **Username**.
7. Configure **Client Management**.

    > **Note**
    > Create a client to use specifically for PingAccess administrative console authentication.

    For more information, see Managing OAuth Clients.
    1. In the **Client Authentication** list, select an option other than **None**.

2. Add the location of the PingAccess host as a **Redirection URI**.

   For example, `https://<PA_Admin_Host>:<PA_Admin_Port>/<reserved application context root>/oidc/cb`.

3. In the **Allowed Grant Type** list, select **Authorization Code**.

4. In the **ID Token Signing Algorithm** list, select one of the elliptic curve (**ECDSA**) algorithms, and in the **Policy** list, select the OIDC policy to use for PingAccess administrative console authentication.

8. To configure **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.

   For more information, see Managing IdP adapter grant mapping.

9. To configure **Access Token Mapping**, on the **Contract Fulfillment** tab, map values into the token attribute contract for the `Username` attribute:

   1. In the **Source** list, select **Persistent Grant**.

   2. In the **Value** list, select **USER_KEY**.

      These are the attributes included or referenced in the access token.

   For more information, see Managing access token mappings.

Next steps

To finish configuring administrator SSO, see Configuring admin UI SSO authentication.

PingOne

Configure PingOne as the token provider for PingAccess.

- Configuring PingOne

Configuring PingOne

Configure PingOne as the token provider in PingAccess.

Before you begin

You must have the PingOne issuer ID, or have access to the PingOne console, to perform this procedure.

Steps

1. Click **Settings**, then go to **System > Token Provider > PingOne**.

2. In the **Issuer** field, enter the PingOne issuer ID.

   This information is available in the PingOne console.

3. **Optional:** In the **Description** field, enter a description for the connection.
4. In the **Trusted Certificate Group** list, select a trusted certificate group for PingAccess to use when authenticating to PingOne.
5. To configure the connection to use a configured proxy, click **Show Advanced Settings** and select **Use Proxy**.

   For more information about creating proxies, see Adding proxies.
6. To configure OAuth 2.0 Demonstrating Proof of Possession (DPoP) settings, click **Show Advanced Settings**:
   1. In the **DPoP Type** list, select the level of DPoP support that you want to enable for access token validation:
      - **Off** (default): PingAccess doesn't accept DPoP-bound access tokens, only bearer tokens.
      - **Enabled**: PingAccess accepts both bearer tokens and DPoP-bound access tokens.
      - **Required**: PingAccess doesn't accept bearer tokens, only DPoP-bound access tokens.
   2. To require each DPoP proof to contain a nonce value during validation that was provided by PingAccess when the access token was created, per RFC 9449 section 9, select **Require Nonce**.

      This check box is cleared by default.
   3. In the **DPoP Proof Lifetime (SEC.)** field, enter the duration, in seconds, that a DPoP proof should be considered valid after it's issued.

      > **Important**
      >
      > As a security best practice, keep this value low and consistent with the DPoP implementation of your API client. The default value is 120 seconds.
7. Click **Save**.

Next steps

After you configure the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **View Metadata → Refresh Metadata**.

PingOne Advanced Identity Cloud or PingAM

Configure PingOne Advanced Identity Cloud or PingAM as the token provider for PingAccess.

- Configuring PingOne Advanced Identity Cloud or PingAM as the token provider

Configuring PingOne Advanced Identity Cloud or PingAM as the token provider

Configure PingOne Advanced Identity Cloud or PingAM as a token provider and OAuth authorization server in PingAccess.

About this task

- Single logout (SLO) isn't supported globally because PingOne Advanced Identity Cloud and PingAM require the **id_token_hint** parameter. The following workaround is available on a per application basis:

    1. Create a virtual logout resource and use the `SLO` and `ID_TOKEN` variables in the **Post-Logout Redirect URL**. For example:

        1. In the PingAccess administrative console, go to **Applications > Applications** and open the associated application.
        2. On the **Resources** tab, click **+ Add Resource**.
        3. In the **Name** field, enter a unique identifier for the resource.
        4. In the **Path Patterns** field, enter the path to which you want to redirect users to initiate logout.

            Redirect users to this logout resource when signing them out instead of the `/pa/oidc/logout` endpoint.

            For example: `/logout`
        5. In the **Resource Authentication** section, click **Standard**.
        6. (Optional) Select the **Audit** checkbox.
        7. In the **Resource Type** list, select **Virtual**.
        8. In the **Response Generators** section, in the **Type** list, select **Logout**.
        9. In the **Post-Logout Redirect URL** field, enter a redirect URL to the token provider's `end_session` endpoint. Include an **id_token_hint**.

            For example: `$(SLO)?id_token_hint=$(ID_TOKEN)`

            You can add more query parameters, such as **post_logout_redirect_uri**, as necessary.
        10. (Optional) Select the **Encode URL** checkbox.
        11. Click **Save**.

    2. Complete the following procedure and make sure to select the **Track id_token** checkbox in step 11. This appends the id token to the associated PingAccess web session.

Steps

1. In the PingAccess administrative console, click **Settings**, then go to **System > Token Provider**. Select **PingOne Advanced Identity Cloud / PingAM**.
2. In the **Issuer** field, enter the PingOne Advanced Identity Cloud or PingAM issuer.
3. (Optional) In the **Description** field, enter a description for the PingOne Advanced Identity Cloud or PingAM token provider.
4. In the **Trusted Certificate Group** list, select the certificate group that the PingOne Advanced Identity Cloud or PingAM certificate is in.

PingAccess requires the certificate in use by the OpenID Connect (OIDC) provider to anchor to a certificate in the associated trusted certificate group.

1. **Java Trust Store** (default): Uses the Java Trust Store for certificate authentication. Learn more in Certificates.
2. **Trust Any**: Allows client authentication with any certificate, including self-signed certificates.

   If you use the **Trust Any** method in production, you should log client certificates in the audit log.

5. In the **Client ID** field, enter the client ID of the OAuth client that will validate the OAuth access tokens.
6. Select a **Client Credentials** Type, then provide the information required for the selected credential type:

   Choose from:

   - Click **Secret** to use a client secret. This is the default selection. In the **Client Secret** field, enter the client secret assigned when you created the PingAccess OAuth client in PingOne Advanced Identity Cloud or PingAM.
   - Click **Mutual TLS** to use Mutual TLS client authentication. In the **Key Pair** list, select a configured key pair to use for Mutual TLS client authentication.
   - Click **Private Key JWT** to use JSON Web Token (JWT). No additional information is required.

7. To retain token details for subsequent requests, in the **Cache Tokens** section, click **Yes**.

   This option reduces communication between PingAccess and PingOne Advanced Identity Cloud or PingAM.

   1. In the **Token Time to Live (Sec.)** field, enter the number of seconds to cache the access token.

      The default value is `-1`, which means no limit.

      > **Important**
      >
      > Enter a value that's less than the PingOne Advanced Identity Cloud or PingAM access token lifetime.

8. In the **Subject Attribute Name** field, enter an attribute from the OAuth access token that you want to track as the subject for auditing purposes.

   At runtime, this attribute's value is used as the subject field in audit log entries for API resources with policies that validate access tokens.

9. To send the URI that the user requested as the PingAccess audience OAuth parameter to PingOne Advanced Identity Cloud or PingAM, select the **Send Audience** checkbox.

   This checkbox is cleared by default.

10. If requests made to PingOne Advanced Identity Cloud or PingAM should use a proxy, click **Show Advanced Settings** and select the **Use Proxy** checkbox.

    This checkbox is cleared by default.

    > **Note**
    >
    > If the node is not configured with a proxy, requests are made directly to the token provider. Learn more about creating proxies in Adding proxies.

11. To track the `id_token` that the authorization server provides after authentication within the PingAccess session cookie, click **Show Advanced Settings** and select the **Track id_token** checkbox.

    This checkbox is cleared by default.

    > **Note**
    >
    > You must select **Track id_token** to use the `id_token` attribute when Creating header identity mappings. You can then use this header to pass along the `id_token` to other Identity mappings, virtual logout resources, or Rules.
    > Token providers can use the `id_token` attribute to identify and locate a user's session. PingOne Advanced Identity Cloud and PingAM require the **id_token_hint** parameter to identify and locate a user's session when performing SLO.

    > **Important**
    >
    > Tracking the `id_token` attribute increases the PingAccess cookie's size. This could make the cookie exceed the browser's limit. Learn more in Minimizing the PingAccess cookie size.

12. Click Save.

Next steps

1. After you configure the token provider, click **View Metadata** to display the metadata provided by the token provider.
2. To update the metadata, click **View Metadata > Refresh Metadata**.

Common token provider

You can configure a common token provider that uses the OpenID Connect (OIDC) protocol as the token provider for PingAccess.

- Configuring OpenID Connect token providers
- Configuring OAuth authorization servers
- Configuring Azure AD as the common token provider when PingAccess is protecting an API application

Configuring OpenID Connect token providers

Configure OpenID Connect (OIDC) token provider settings in PingAccess.

Steps

1. Click **Settings**, then go to **System > Token Provider > Common > OpenID Connect**.
    1. Go to **Settings > System > Token Provider** and select **Common Token Provider**.
2. In the **Issuer** field, enter the OIDC provider's issuer identifier.
3. **Optional:** In the **Description** field, enter a description for the token provider.

4. To record requests to the OIDC provider to the audit store, select the **Audit** check box.
5. If required, click **+ Add Query Parameter** and enter custom query parameter name and value pairs used by the OIDC provider.
6. In the **Trusted Certificate Group** list, select the group of certificates to use when authenticating to the OIDC provider.

   PingAccess requires the certificate in use by the OIDC provider to anchor to a certificate in the associated Trusted Certificate Group.
7. To configure advanced settings, click **Show Advanced**.
   1. 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. See Adding proxies for more information about creating proxies.

   2. Select the **Use Single-Logout** check box to enable single logout (SLO) when the `/pa/oidc/logout/` endpoint receives a request to clear the cookie containing the PingAccess token.

      If you select this option, PingAccess sends a logout request to the token provider after receiving a request at the `/pa/oidc/logout/` endpoint. The token provider then completes a full SLO flow.

      > **Note**
      >
      > To use this feature, you must configure SLO on the OIDC provider.

   3. Select the **Track id_token** check box to track the `id_token` that the authorization server provides after authentication within the PingAccess session cookie.

      > **Note**
      >
      > Token providers can use the `id_token` attribute to identify and locate a user's session. Some token providers may require an `id_token_hint` parameter for SLO, but not all. For more information on this configuration, see the table entry **Include id_token_hint in SLO** in step 8 of Configuring admin UI SSO authentication.

      You must select **Track id_token** to use the `id_token` attribute when Creating header identity mappings. You can then use this header to pass along the `id_token` to other Identity mappings or Rules.

      > **Important**
      >
      > Tracking the `id_token` attribute increases the PingAccess cookie's size. This could make the cookie exceed the browser's limit. For more information, see Minimizing the PingAccess cookie size.

   4. Select **Request Supported Scopes Only** to limit the requested scopes to those advertised in the OIDC metadata.
8. Click **Save**.

Next steps

After 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**.

Creating Azure AD Graph API applications

To use the Azure AD Graph application programming interface (API), an application must exist to provide an application ID and key that PingAccess will use as the client ID and client secret for communication with the Graph API.

About this task

Create the application in Azure AD through the **App Registrations** blade using these criteria:

Name

Enter a unique name for the application, such as "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.

Steps

1. After you create the application, navigate to the application in the list.
2. Select **Required permissions** and click **Add**.
3. Choose **Windows Azure Active Directory**, and then click **Save**.

   For **Application Permissions**, read the directory data.
4. Copy the **Application ID**.
5. Generate and copy a **Key**.

Configuring token provider-specific options

Configure plugins that perform particular functions for the selected token provider type.

Before you begin

In order to configure these options, you must first perform the steps detailed in Creating Azure AD Graph API applications.

About this task

In the case of the PingAccess for Azure AD solution, the plugin addresses the following problems:

- Data Transformation— The format of data returned from the OpenID Connect (OIDC) UserInfo endpoint results in some unexpected JavaScript Object Notation (JSON) formatting. This data transforms into a format that PingAccess can easily process.
- Azure AD Graph application programming interface (API) usage— If the **groups** attribute contains more than 200 groups, the id_token contains a level of indirection that points to a Uniform Resource Locator (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.

- Retrieving 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 Azure AD 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.

Steps

1. Click **Settings**, then go to **System > Token Provider > Common > OpenID Connect**.
    1. Go to **Settings > System > Token Provider** and select **Common Token Provider**.
2. Go to **Token Provider Specific Options** section.
3. From the **Type** list, select **Azure Active Directory**.
4. To extend the attributes for a web session, select the **Use Azure AD Graph API** check box.
5. In the **Client ID** field, enter the application ID you copied from the Azure AD API application you created.
6. In the **Client Secret** field, paste the key you copied. Select **Retrieve Group Display Names**.

> **Important**
>
> To retrieve group 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` API, select the following permission, and enter and specify a group type, such as `SecurityGroup` .

7. Select **Cache Group Display Names** to instruct PingAccess to cache display names retrieved from the Azure AD Graph API.
8. In the **Display Name Cache Max Age (s)** field, enter the number of seconds to cache group display names if caching is enabled. Click **Save**.

Configuring OAuth authorization servers

Configure, modify, and edit the OAuth authorization servers in PingAccess.

Before you begin

If you plan to use **Mutual TLS**, modify the token provider to provide the `mtls_endpoint_aliases` object, with content defined by RFC-8705, on the OpenID Connect (OIDC) well-known configuration endpoint.

Steps

1. Click **Settings**, then go to **System > Token Provider > Common > OAuth Authorization Server**.
2. **Optional:** In the **Description** field, enter a description for the authorization server.
3. In the **Targets** field, enter one or more `hostname:port` pairs for the OAuth authorization server.
    1. **Optional:** Click **+ Add Target** to add additional targets.

4. In the **Introspection Endpoint** field, specify the OAuth endpoint through which the token introspection operation is accomplished.

   > **Note**
   >
   > If you've configured a remote token access validator on your PingAccess application and try to remove the **Introspection Endpoint** or save without configuring it on the **OAuth Authorization Server** tab, you get the following error message:
   >
   > ```
   > Introspection endpoint is required as there are applications that use remote token validation.
   > ```
   >
   > Remove the remote access token validator before filling out your configuration on the **OAuth Authorization Server** tab.

5. In the **Token Endpoint** field, enter the OAuth 2.0 Authorization Server's token endpoint.
6. To record requests to the OAuth authorization server to the audit store, select the **Audit** check box.
7. If the OAuth authorization server is expecting HTTPS connections, select the **Secure** option.
8. In the **Trusted Certificate Group** list, select the group of certificates to use when authenticating to the OAuth authorization server.

   PingAccess requires that the certificate in use by the OAuth authorization server anchors to a certificate in the associated trusted certificate group.

9. In the **Client ID** field, enter the unique identifier assigned when you created the PingAccess OAuth client within your OAuth authorization server.
10. Select a **Client Credentials Type**, then provide the information required for the selected credential type.

    Choose from:

    - **Secret**: Enter the **Client Secret** assigned when you created the PingAccess OAuth client in the token provider.
    - **Mutual TLS**: Select a configured **Key Pair** to use for mutual TLS client authentication.
    - **Private Key JWT**: Select this option to use Private Key JSON web token (JWT). No additional information is required.

11. **Optional:** To retain token details for subsequent requests, select the **Cache Tokens** option.

    This option reduces communication between PingAccess and the OAuth authorization server.

12. **Optional:** To enter the number of seconds to cache the access token, select the **Token Time To Live** check box.

    A value of -1 means that there is no limit. This value should be less than the OAuth authorization server token lifetime.

13. In the **Subject Attribute Name** field, enter the attribute that 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 application programming interface (API) resources with policies that validate access tokens.

14. To send the URI the user requested as the `aud` OAuth parameter for PingAccess to the OAuth 2.0 authorization server, select the **Send Audience** check box.

15. To configure the connection to use a configured proxy, click **Show Advanced Settings** and select **Use Proxy**.

    For more information about creating proxies, see Adding proxies.

16. To configure OAuth 2.0 Demonstrating Proof of Possession (DPoP) settings, click **Show Advanced Settings**:

    1. In the **DPoP Type** list, select the level of DPoP support that you want to enable for access token validation:
       - **Off** (default): PingAccess doesn't accept DPoP-bound access tokens, only bearer tokens.
       - **Enabled**: PingAccess accepts both bearer tokens and DPoP-bound access tokens.
       - **Required**: PingAccess doesn't accept bearer tokens, only DPoP-bound access tokens.
    2. To require each DPoP proof to contain a nonce value during validation that was provided by PingAccess when the access token was created, per RFC 9449 section 9, select **Require Nonce**.

       This check box is cleared by default.
    3. In the **DPoP Proof Lifetime (SEC.)** field, enter the duration, in seconds, that a DPoP proof should be considered valid after it's issued.

       > **Important**
       >
       > As a security best practice, keep this value low and consistent with the DPoP implementation of your API client. The default value is 120 seconds.

17. Click **Save**.

> **Note**
>
> If the node isn't configured with a proxy, requests are made directly to the token provider.

Configuring Azure AD as the common token provider when PingAccess is protecting an API application

Configure Microsoft Azure AD to mint access tokens that can be validated locally when PingAccess is protecting an API application.

Before you begin

Make sure that you've configured an application in Microsoft Azure AD. If you haven't, see creating Azure AD Graph API applications.

About this task

If you're using PingAccess to protect an API application and want to use Azure AD as the common token provider, you must complete this task. Because Microsoft Azure AD doesn't have an `introspection` endpoint to validate access tokens remotely, you must use a key from the JSON Web Key Set (JWKS) endpoint to validate access tokens locally. This task prevents Azure AD from adding a nonce value to the access token after it's been signed because adding the nonce value blocks PingAccess's ability to validate the access token.

However, if you're protecting a web application with PingAccess and want to configure Azure AD as the common token provider, see configuring token provider-specific options instead. You don't need to complete this task when PingAccess is protecting a web application because the `userinfo` endpoint in Azure AD can use the nonce value that Azure AD inserts into the access token and validate the access token remotely.

To configure Azure AD to mint an access token that can be validated locally:

Steps

1. In your Azure AD environment, create a scope for the API application that you want to protect:
    1. On the **Overview** page of your Azure AD application, go to the **Essentials** section and copy the **Application (client) ID**.
    2. Go to **Manage → Expose an API** and in the **Scopes** section, click **Add a scope**.

       Use the following format to create the scope for your API application:

       ```
       api://<Application (client) ID>/<scope name>
       ```

       Make sure to paste in the **Application (client) ID** that you copied in step 1a.

       > **Important**
       >
       > This scope is what prevents Azure AD from minting an access token with a nonce value that's only usable by the `userinfo` endpoint.

2. In the PingAccess administrative console, set up Azure AD as a common token provider:
    1. Go to **Settings > System > Token Provider** and select **Common Token Provider**.
    2. Fill out the **OpenID Connect** tab according to the configuring PingAccess to use Azure AD as the token provider and configuring token provider-specific options procedures.
    3. Fill out the **OAuth Authorization** tab according to the configuring OAuth authorization servers procedure.

       Troubleshooting:

       > **Note**
       >
       > If you've configured a remote token access validator on your PingAccess application and try to remove the **Introspection Endpoint** or save without configuring it on the **OAuth Authorization Server** tab, you get the following error message:
       >
       > ```
       > Introspection endpoint is required as there are applications that use remote token validation.
       > ```
       >
       > Remove the remote access token validator before filling out your configuration on the **OAuth Authorization Server** tab.

3. In your PingAccess application, set up a JWKS endpoint validator to perform local validation in Azure AD:
    1. Go to **Applications > Applications**, click the **Expand** icon to view more details about the API application that you want to protect, then click the **Pencil** icon.

2. In the **Application Type** section, click **+ Create** below the **Access Validation** list.

   Do not select the remote access token validator, **Token Provider**, in the **Access Validation** list. You must instead set up a local access token validator.

   > **Note**
   >
   > If you try to use a remote access token validator on your PingAccess API application without first configuring the introspection endpoint on the **OAuth Authorization Server** tab of the **Token Provider** page, you get the following error message:
   >
   > ```
   > Cannot use remote validation as authorization server does not have an Introspection endpoint.
   > ```

3. In the **Name** field, enter a name for the token validator.
4. In the **Type** list, select **JSON Web Key Set (JWKS) Endpoint**.
5. In your Azure AD application, go to the **Endpoints** tab on the **Overview** page and copy the full URL of the `OIDC metadata document` endpoint.
6. Open the OpenID Connect (OIDC) metadata document URL in a new tab and copy the value of the `jwks_uri` property beginning after the https://login.microsoft.com segment.

   Example:

   <Directory (tenant) ID>/discovery/v2.0/keys

7. In PingAccess, return to the **New Access Token Validator** page and paste the `jwks_uri` value that you copied into the **Path** field.
8. Click **Save**.

4. Add the scope that you set up in Azure AD to your PingAccess web session:

   1. In your Azure AD application, go to **Manage → Expose an API** and in the **Scopes** section, copy the full path of the scope that you set up in step 1b.
   2. In PingAccess, go to **Access > Web Sessions**, click the **Expand** icon to view more details about the web session associated with your API application, then click the **Pencil** icon.
   3. Click **Show Advanced Settings**, then in the **Scopes** field, paste the full path of the scope that you copied in step 4a.
   4. Click **Save**.

Result

You now have an access token that PingAccess can validate and have finished configuring your PingAccess application, web session, and access token validator to use Azure AD as the common token provider.

Environment

View and update the details of your PingAccess environment.

You can manage information about your PingAccess environment, such as the environment name, displayed in the upper right section of the user interface. For more information, see Changing the Environment Name.

Changing the Environment Name

Change the environment name displayed in the PingAccess user interface.

About this task

The environment name is displayed in the upper right header bar. If you manage multiple instances of PingAccess, or more than one Ping product, use this environment name to delineate these environments.

Steps

1. Click **Settings**, then go to **System > Environment**.
2. In the **Environment Name** field, enter an environment name.
3. Click **Save**.

   Result:

   The new environment name is displayed in the header bar.

System defaults

View and update the details of your PingAccess system defaults.

You can manage information about your PingAccess default authentication challenge policy. For more information, see Changing the default authentication challenge policy.

Changing the default authentication challenge policy

Change the default authentication challenge policy in the PingAccess application defaults.

About this task

You can change the default authentication challenge policy so that when you create new applications for `Web` or `Web+API`, SPA support is automatically disabled.

Steps

1. Click **Settings**, then go to **System > System Defaults**.
2. In the **Default Authentication Challenge Policy** list, select an Authentication Challenge Policy to produce authentication challenges for the application.
3. Click **Save**.

   Result:

   The specified **Default Authentication Challenge Policy** will be the default for new `Web` or `Web+API` applications in the PingAccess Admin UI.

Managing PingOne connections

Add a credential from your PingOne environment to the PingAccess administrative console to establish a connection between PingAccess and PingOne Protect.

About this task

PingOne connections are currently only used as part of PingAccess's integration with PingOne Protect. For a more detailed explanation of this integration, see PingOne Protect integration.

You must create a PingOne connection before you can create a risk policy in PingAccess. A PingOne connection makes it possible for PingAccess to receive risk evaluations from PingOne Protect.

To create or manage PingOne connections through the PingAccess administrative console, see:

Adding a PingOne connection

Steps

1. Configure PingOne Protect for connectivity with PingAccess:
    1. In your PingOne administrative environment, go to **Integrations → PingFederate**.

       Currently, you must use a PingFederate connection because PingAccess does not have one of its own.
    2. Expand an existing connection or click the **+** icon to create a new one.
    3. On the **Overview** tab, click the **+** icon next to **Credentials**.
    4. Click the **Copy to clipboard** icon, then click **Done**.

       Result:

       This copies a JSON Web Token (JWT) to your clipboard that contains information on your PingOne environment.
2. Configure PingAccess for connectivity with PingOne Protect:
    1. In PingAccess, go to **Settings → System → PingOne Connections** and click **+Add PingOne Connection**.
    2. Complete the fields.

       For more information, see PingOne connection field descriptions.
    3. Click **Save**.

Next steps

After you've created a connection, you can assign it to a specific risk policy through the **Risk Policies** page. For more information, see Adding a risk policy.

Editing a PingOne connection

Steps

1. Go to **Settings → System → PingOne Connections**.

2. Click the **Expand** icon to view more details about the connection that you want to edit.
3. On the **Properties** tab, click the **Pencil** icon.
4. Make the required changes.

   For more information, see PingOne connection field descriptions.
5. Click **Save**.

Deleting a PingOne connection

Steps

1. Go to **Settings → System → PingOne Connections**.
2. Click the **Expand** icon to view more details about the connection that you want to delete.
3. Click the **Delete** icon.
4. Click **Delete**.

PingOne connection field descriptions

The following table describes the fields available for managing PingOne connections on the **PingOne Connections** tab in the PingAccess administrative console.

Field	Required	Description
Name	Yes	A unique name for this PingOne connection.
Description	No	An optional description for this PingOne connection.
Credential	Yes	In the **Credential** field, paste the JWT that you copied to your clipboard in step 1d of Adding a PingOne connection. This credential is necessary to establish a connection with PingOne. **Note** Whenever you save a PingOne connection, PingAccess validates the credential you specified to make sure it can communicate with the associated PingOne environment.

Advanced Settings

To configure advanced settings on a PingOne connection, expand the **Show Advanced Settings** section at the bottom of the **PingOne Connections** tab. These settings are optional.

Field	Description
Trusted Certificate Group	The group of certificates that you want to use to establish trust when using this connection to communicate with PingOne.
Use Proxy	Select the **Use Proxy** check box if you plan to use a proxy to send HTTPS requests to PingOne.

Log settings

Use the **Log Settings** page to customize your PingAccess logs so that you can pinpoint and resolve issues in a more convenient manner.

The **Verbose** log setting enables you to adjust the amount of information available in your log entries for PingAccess-specific logging categories without needing to apply changes directly within the `conf/log4j2.xml` file.

> **Tip**
>
> PingAccess records runtime, administrative server activities, and audit activity in the respective log files contained in the `<PA_HOME>/log` folder. For more information on logging in PingAccess, see the Log configuration section.

You can adjust log levels with the **Verbose** log setting through the administrative console or the administrative API. For more information, see:

- Learn more about configuring log levels in the administrative console in Configuring verbose logging in the admin console. Learn more about configuring log levels in the administrative API in the admin API documentation at https://`<host>`:`<admin-port>`/pa-admin-api/v3/api-docs/.
- Learn more about the default categories in Log level category descriptions.
- Learn more about creating custom log level categories in Creating custom log level categories.

Configuring verbose logging in the admin console

Select or clear the **Verbose** log setting check box for each PingAccess-specific logging category to adjust log levels more conveniently and quickly.

Before you begin

If you have customized your `log4j2.xml` file, you must merge this file with the `log4j-categories.xml` file introduced in PingAccess 7.3. Otherwise, you receive a warning message after opening the **Log Settings** page. If you receive this warning, it means that your changes have been saved, but can't be implemented until you have merged the categories that you referenced into the `log4j2.xml` file.

About this task

Logging levels, sometimes abbreviated to log levels, control the amount of information displayed in each log entry for a specific category. The `log4j-categories.xml` file, first introduced in PingAccess 7.3, defines and allows references to PingAccess-specific logging categories. The **Verbose** log setting enables you to adjust log levels in the admin console as opposed to directly within the `conf/log4j2.xml` file. Any changes you make with this setting are effective immediately.

> **Tip**
>
> PingAccess records runtime, administrative server activities, and audit activity in the respective log files contained in the `<PA_HOME>/log` folder. For more information on logging in PingAccess, see the Log configuration section.

Steps

1. In PingAccess, go to **Settings → System → Log Settings**.
2. In the **Verbose** column, select the check boxes that correspond with the categories you want to change.

   By default, all categories (except for the five audit categories) are set to the `INFO` level. Enabling verbose logging changes the log level for the designated category to `DEBUG` or `TRACE`, respectively. When verbose logging is turned off for a designated category, the log level returns to `INFO`.

   > **Important**
   >
   > Leaving log categories set to `DEBUG` or `TRACE` can decrease your server performance, so make sure to turn off verbose logging when you've finished reviewing your logs.

   For more information on the categories that are available for you to adjust, see Log level category descriptions.
3. Click **Save**.

Log level category descriptions

The following table describes the fields available for managing log categories on the **Log Settings** page and within the `/logSettings` endpoint.

> **Note**
>
> You can customize the log categories in both the admin console and the admin API by editing the `log4j-categories.xml` file directly. For more information, see the comments in the file.

Field	Description	Might Log Sensitive Information if Enabled
Core	Provides debug logging for core components.	No

Field	Description	Might Log Sensitive Information if Enabled
Cluster Replication	Logs configuration replication details.	No
HTTP Request Headers	Logs HTTP request headers.	Yes
HTTP Request Parameters	Logs HTTP `GET` request parameters.	Yes
HTTP Client	Provides information about requests and responses that clients made to PingAccess.	Yes
HTTP Application	Provides information about requests and responses that PingAccess made to other tools or services.	Yes
Cookie	Logs both incoming and outgoing cookies.	No
Network Events	Analyzes the state of connections to PingAccess.	No
Configuration Management	Logs details related to loading the administrative configuration from the disk to the runtime.	No
CRL Handling	Logs certificate revocation checking and validation.	No
Groovy Rule	Logs information from within a groovy rule script.	No
API Audit	Logs API audit traffic, including requests and responses. This category is enabled by default.	No
Engine Audit	Logs engine audit traffic, including requests and responses. This category is enabled by default.	No
Agent Audit	Logs agent audit traffic, including requests and responses. This category is enabled by default.	No
Sideband Client Audit	Logs sideband client audit traffic, including requests and responses. This category is enabled by default.	No

Field	Description	Might Log Sensitive Information if Enabled
Sideband Audit	Logs sideband client audit traffic, including requests and responses. This category is enabled by default.	No

Creating custom log level categories

Create a custom log category to track specific troubleshooting information in PingAccess, if it's available for collection.

Steps

1. In the `<pa_home>/conf/log4j-categories.xml` file, go to the **logCategories** section and add a new category entry.

   Use the following syntax: `<category id="" name="" offLevel="" onLevel="" description="">` where:

   category id

   Is the logger's reference name. This value must be a unique alphanumeric string.

   > **Tip**
   >
   > You will use the **category id** in step 3.

   name

   Is the logger's nickname. This value appears in the admin console, so it should be descriptive enough to identify the logger.

   offLevel

   Is the default log level that's used when the category is deselected in the admin console. Valid values are:

   - FATAL
   - ERROR
   - WARN
   - INFO (recommended)
   - DEBUG
   - TRACE

   > **Tip**
   >
   > You will use the **offLevel** in step 3.

onLevel

Is the log level that's used to troubleshoot issues when the category is selected in the admin console. Valid values are:

- FATAL
- ERROR
- WARN
- INFO
- DEBUG (recommended)
- TRACE

description

Is the unique description for the logger. This description appears in the admin console.

2. Save and close the `log4j-categories.xml` file.
3. Open the `<pa_home>/conf/log4j2.xml` file, go to the `Loggers` section, and add a logger entry.

   Use the following syntax: `<Logger name="" level="${sys:pa.log.level.<categoryid>:-<offLevel>}"/>` where:

sys:pa.log.level

Is a constant value for all logger entries.

<categoryid>

Is the **id** value from the category entry. This value is case sensitive.

<offLevel>

Is the default starting logging level.

> **Important**
>
> This value should match the **offLevel** value in the category entry.

4. Save and close the `log4j2.xml` file.
5. If you are running PingAccess in clustered mode, copy both files to each PingAccess instance in the cluster that you want to apply the settings to.
6. Restart PingAccess.

Result

The **Log Settings** page displays the new categories, and you can select or deselect those categories as described in Configuring verbose logging in the admin console.

Agents and Integrations

PingAccess is supported by agents that can be installed in an agent deployment, and integrations that help PingAccess work with other tools.

Agents

In an agent deployment, agents are installed on each web server. You can use an existing agent or create your own using the SDKs.

- PingAccess Agent for Apache (RHEL)
- PingAccess Agent for Apache (SLES)
- PingAccess Agent for Apache (Windows)
- PingAccess Agent for IIS
- PingAccess Agent for NGINX
- PingAccess agent protocol
- PingAccess Agent SDK for C
- PingAccess Agent SDK for Java

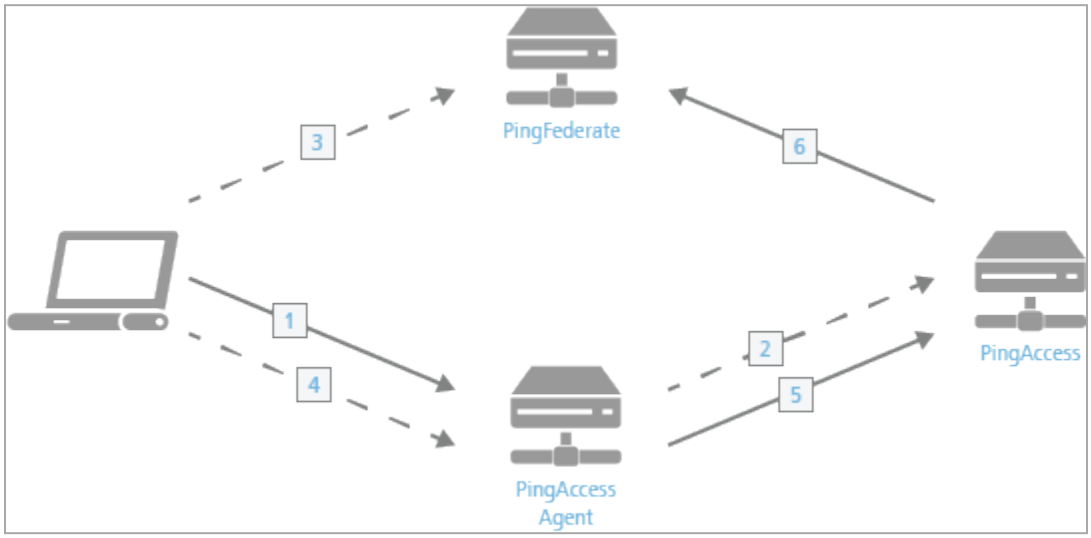
Integrations

You can create integrations for PingAccess using the Add-on Software Development Kit (SDK), or use existing integrations with products used in your environment.

- PingAccess Add-on SDK for Java
- iovation Device Risk Integration
- IWA Integration
- Kong API Gateway Integration
- Apigee API Gateway Integration
- PingOne Protect integration

PingAccess Agent for Apache (RHEL)

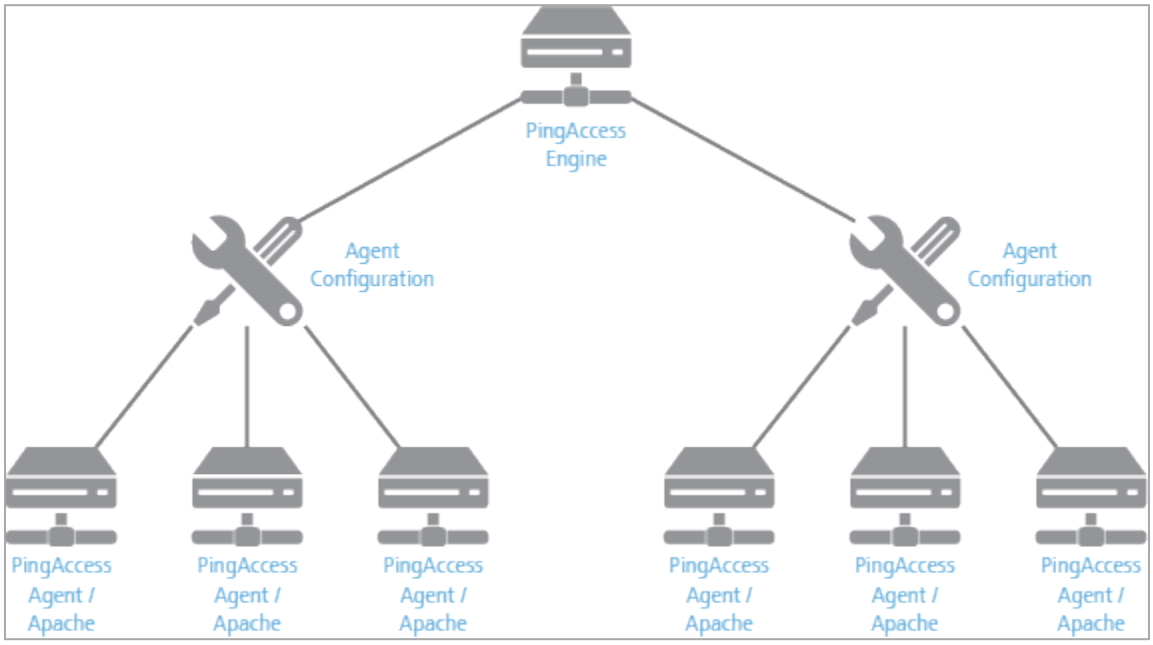
The PingAccess Agent for Apache is an Apache module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



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 signs on, 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 administrative 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.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

Installing on RHEL

Install or uninstall a PingAccess agent on a RHEL system.

Before installing the agent, review:

- RHEL agent system requirements.

To install the PingAccess agent for Apache:

- For information on RHEL 8, learn more in Installing on RHEL 8.
- For information on RHEL 9, learn more in Installing on RHEL 9.
- For information on IBM HTTP Server 9.0, learn more in Installing on an IBM HTTP Server using Apache 2.4.
- For information on IBM HTTP Server 8.5.5, learn more in Installing on an IBM HTTP Server using Apache 2.2.

To uninstall the agent:

- Learn more in Uninstalling the RHEL agent.

RHEL agent system requirements

The PingAccess agent for Apache (RHEL) is supported on the following platforms:

HTTP Server	RHEL Server
Apache HTTP Server 2.4	• Running on Red Hat Enterprise Linux Server 8 (x86_64) • Running on Red Hat Enterprise Linux Server 9 (x86_64)
IBM HTTP Server 8.5.5 **Note** The latest supported PingAccess agent for this server is the PingAccess agent for Apache (RHEL) 1.6.0. It must be installed with the Apache HTTP Server 2.2 running on Red Hat Enterprise Linux Server 8 (x86_64). Learn more in Installing on an IBM HTTP Server using Apache 2.2.	• Running on Red Hat Enterprise Linux Server 8 (x86_64)
IBM HTTP Server 9.0	• Running on Red Hat Enterprise Linux Server 8 (x86_64) • Running on Red Hat Enterprise Linux Server 9 (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. You should deploy SELinux on the server.

Installing on RHEL 8

Install a PingAccess agent on a RHEL 8 system with Apache 2.4, either conventionally or manually.

Before you begin

If you haven't downloaded an `agent.properties` file, you must first:

1. In the PingAccess console, go to **Applications → Agents**.
2. Click the **Pencil** icon to edit a configured agent.

   If you haven't created an agent yet, see Agents.
3. In the **Shared Secrets** section, click the **Download** icon to download the configuration.

   Note

   The configuration file will be named `<agentname>_agent.properties` .

About this task

Click the respective tab for the installation that you want to use.

Installing on RHEL 8

Installing on RHEL 8 conventionally
Before you begin

Download and extract the `pingaccess-agent-apache24-rhel8-<version>.zip` archive.

> **Note**
>
> The Agent RPM has required dependencies that might be available through standard repositories. If these dependencies are not available in your Linux version, you can install them using the included `openpgm-5.2.122-21.el8.x86_64.rpm`, `libsodium-1.0.18-2.el8.x86_64.rpm`, `libunwind-1.3.1-3.el8.x86_64.rpm`, and `zeromq-4.3.2-1.el8.x86_64.rpm` packages.
> You can install these RPMs using the following command:
>
> ```
> yum install libsodium*rpm openpgm*rpm libunwind*rpm zeromq*rpm
> ```

About this task

To install a PingAccess agent on a RHEL 8 system with Apache 2.4:

Steps

1. In RHEL, change to the `pingaccess-agent-apache24-rhel8-<version>/x86_64` directory.
2. As root, install the PingAccess Agent for Apache using the following command:

   ```
   yum install pingaccess-agent-apache-*.rpm
   ```

3. Copy the `<agentname>_agent.properties` file to `/etc/httpd/conf.d/agent.properties`.
4. As root, restart the Apache service using the following command:

   ```
   systemctl restart httpd
   ```

Manually Installing on RHEL 8

Manually Installing on RHEL 8
About this task

This procedure assumes that:

- A non-root user is installing the PingAccess Agent for Apache in a custom Apache instance.
- You've installed the Apache installation at `$APACHE`. If you haven't, modify the file paths specified in this procedure based on where your Apache installation and configuration files are located.

To manually install the PingAccess Agent for Apache on a RHEL 8 system 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` archive into a directory called `pkgroot` and unpack them using the following commands:

```
mkdir pkgroot
cp *.rpm pkgroot/
cd pkgroot
for r in *.rpm; do rpm2cpio $r | cpio -idmv; done
```

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:
    1. 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
```

    2. Change all occurrences of `conf.d` to `conf`.

6. Copy your downloaded `<hostname>_agent.properties` to `$APACHE/conf/agent.properties`.
7. Run the command `$APACHE/bin/apachectl restart` to restart Apache.

Installing on RHEL 9

Install a PingAccess agent on a RHEL 9 system with Apache 2.4, either conventionally or manually.

Before you begin

If you haven't downloaded an `agent.properties` file, you must first:

1. In the PingAccess console, go to **Applications → Agents**.
2. Click the **Pencil** icon to edit a configured agent.

   If you haven't created an agent yet, see Agents.
3. In the **Shared Secrets** section, click the **Download** icon to download the configuration.

> **Note**
>
> The configuration file will be named `<agentname>_agent.properties`.

About this task

Click the respective tab for the installation that you want to use.

Installing on RHEL 9

Installing on RHEL 9 conventionally

Before you begin

Download and extract the `pingaccess-agent-apache24-rhel9-<version>.zip` archive.

> **Note**
>
> The Agent RPM has required dependencies that might be available through standard repositories. If these dependencies aren't available in your Linux version, you can install them using the included `openpgm-5.2.122-28.el9.x86_64.rpm`, `libsodium-1.0.18-8.el9.x86_64.rpm`, `libunwind-1.6.2-1.el9.x86_64.rpm`, and `zeromq-4.3.4-2.el9.x86_64.rpm` packages.
> You can install these RPMs using **`yum install libsodium*rpm openpgm*rpm libunwind*rpm zeromq*rpm`**.

About this task

To install a PingAccess agent on a RHEL 9 system with Apache 2.4:

Steps

1. In RHEL, change to the `pingaccess-agent-apache24-rhel9-<version>/x86_64` directory.
2. As root, install the PingAccess Agent for Apache using the following command:

   ```
   yum install pingaccess-agent-apache-*.rpm
   ```

3. Copy the `<agentname>_agent.properties` file to `/etc/httpd/conf.d/agent.properties`.
4. As root, restart the Apache service using the following command:

   ```
   systemctl restart httpd
   ```

Manually Installing on RHEL 9

Manually Installing on RHEL 9
About this task

This procedure assumes that:

- A non-root user is installing the PingAccess Agent for Apache in a custom Apache instance.
- You've installed the Apache installation at `$APACHE`. If you haven't, modify the file paths specified in this procedure based on where your Apache installation and configuration files are located.

To manually install the PingAccess Agent for Apache on a RHEL 9 system when Apache is installed in a non-standard way:

Steps

1. Install the following required dependencies from the Red Hat 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
for r in *.rpm; do rpm2cpio $r | cpio -idmv; done
```

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. Edit the `10-paa.conf` file and make the following changes:
    1. Add the following lines before the `LoadModule` directive:

```
LoadFile modules/libunwind.so.8
LoadFile modules/libpgm-5.2.so.0
LoadFile modules/libsodium.so.23
LoadFile modules/libzmq.so.5
```

    2. Change all occurrences of `conf.d` to `conf`.
5. Copy your downloaded `<hostname>_agent.properties` to `$APACHE/conf/agent.properties`.
6. Run the `$APACHE/bin/apachectl restart` command to restart Apache.

Installing on an IBM HTTP Server using Apache 2.4

Manually install a PingAccess agent on a RHEL system with Apache 2.4 when using an IBM HTTP Server.

Before you begin

If you haven't downloaded an `agent.properties` file:

1. In the PingAccess admin console, go to **Applications > Agents**.
2. Click the **Pencil** icon to edit a configured agent.

   If you haven't created an agent yet, learn more about how to do so in Adding agents.
3. In the **Shared Secrets** section, click the **Download** icon to download the configuration.

> **Note**
>
> The configuration file will be named `<agentname>_agent.properties` .

This procedure assumes that:

- You've installed and configured the IBM HTTP Server according to IBM's documentation.
- You've downloaded and extracted the version-appropriate `.zip` archive for your environment. For example:
  - `pingaccess-agent-apache24-rhel8*.zip`
  - `pingaccess-agent-apache24-rhel9*.zip`
- `apachectl` for the running IBM HTTP Server instance is in the file path.
- You've installed the Apache installation at `$IHS`. If you haven't, modify the file paths specified in this procedure based on where your Apache installation and configuration files are located.
- You've installed `libcurl` and PCRE or verified that they are installed. To install these packages, use the `yum install libcurl pcre` command.

About this task

Click the tab for the installation that you want to use.

With RHEL 8

Installing on a RHEL 8 system with an IBM HTTP Server
About this task

To manually install the PingAccess agent for Apache on RHEL 8 when using the IBM HTTP Server:

Steps

1. Go to the `pingaccess-agent-apache24-rhel8-<version>/<arch>/` directory.

   > **Note**
   > Currently, the only valid value for *<arch>* is `x86_64` for 64-bit.

   Example:

   ```
   cd pingaccess-agent-apache24-rhel8-3.0.0/x86_64/
   ```

2. Extract the package RPMs using the following command:

   ```
   mkdir pkgroot
   cp *.rpm pkgroot/
   cd pkgroot
   for r in *.rpm; do rpm2cpio $r | cpio -idmv; done
   ```

3. Run the **`cp`** command to copy the libraries to the appropriate Apache directories.

   Example:

   For RedHat Enterprise Linux 8 (x86_64):

   ```
   cp -av usr/lib64/.so $IHS/modules
   ```

4. Copy `mod_paa.so` into the Apache modules directory:

   ```
   cp -av usr/lib64/httpd/modules/mod_paa.so $IHS/modules
   ```

5. Copy the `10-paa.conf` file to the Apache configuration directory:

   ```
   cp -av etc/httpd/conf.modules.d/10-paa.conf $IHS/conf
   ```

6. In the `10-paa.conf` file:

   1. Add the following lines before the `LoadModule` directive:

      ```
      LoadFile modules/libpgm-5.2.so.0
      LoadFile modules/libzmq.so.5
      ```

2. Update the values for `PaaPropertyFiles` and `PaaCertificateDir` to point to your Apache `conf` directory.

7. In the Apache configuration file, `$IHS/conf/httpd.conf`, use the following directive to add the PingAccess agent for Apache's module configuration:

```
Include conf/10-paa.conf
```

8. Copy the `<agentname>_agent.properties` file to the `$IHS/conf/agent.properties` directory.

   This is the configuration file that you downloaded in step 3 of Installing on an IBM HTTP Server using Apache 2.4.

9. Restart the Apache service by running the **`apachectl restart`** command.

With RHEL 9

Installing on a RHEL 9 system with an IBM HTTP Server
About this task

To manually install the PingAccess agent for Apache on RHEL 9 when using the IBM HTTP Server:

Steps

1. Go to the `pingaccess-agent-apache24-rhel9-<version>/<arch>/` directory.

   > **Note**
   > Currently, the only valid value for *<arch>* is `x86_64` for 64-bit.

   Example:

   ```
   cd pingaccess-agent-apache24-rhel9-3.0.0/x86_64/
   ```

2. Extract the package RPMs using the following command:

   ```
   mkdir pkgroot
   cp *.rpm pkgroot/
   cd pkgroot
   for r in *.rpm; do rpm2cpio $r | cpio -idmv; done
   ```

3. Run the **`cp`** command to copy the libraries to the appropriate Apache directories.

   Example:

   For RedHat Enterprise Linux 9 (x86_64):

   ```
   cp -av usr/lib64/.so $IHS/modules
   ```

4. Copy `mod_paa.so` into the Apache modules directory:

   ```
   cp -av usr/lib64/httpd/modules/mod_paa.so $IHS/modules
   ```

5. Copy the `10-paa.conf` file to the Apache configuration directory:

   ```
   cp -av ../10-paa.conf $IHS/conf
   ```

6. In the `10-paa.conf` file:

   1. Add the following lines before the `LoadModule` directive:

      ```
      LoadFile modules/libpgm-5.2.so.0
      LoadFile modules/libzmq.so.5
      ```

2. Update the values for `PaaPropertyFiles` and `PaaCertificateDir` to point to your Apache `conf` directory.

7. In the Apache configuration file, `$IHS/conf/httpd.conf`, use the following directive to add the PingAccess agent for Apache's module configuration:

```
Include conf/10-paa.conf
```

8. Copy the `<agentname>_agent.properties` file to the `$IHS/conf/agent.properties` directory.

   This is the configuration file that you downloaded in step 3 of Installing on an IBM HTTP Server using Apache 2.4.

9. Restart the Apache service by running the **`apachectl restart`** command.

Installing on an IBM HTTP Server using Apache 2.2

Manually install a PingAccess agent on a RHEL system with Apache 2.2 when using IBM HTTP Server 8.5.5.

Before you begin

If you haven't downloaded an `agent.properties` file:

1. In the PingAccess admin console, go to **Applications > Agents**.
2. Click the **Pencil** icon to edit a configured agent.

   If you haven't created an agent yet, learn more about how to do so in Adding agents.

3. In the **Shared Secrets** section, click the **Download** icon to download the configuration.

> **Note**
>
> The configuration file will be named `<agentname>_agent.properties`.

This procedure assumes that:

- You've installed and configured the IBM HTTP Server according to IBM's documentation.
- You've downloaded and extracted the version-appropriate `.zip` archive for your environment. For example:
  - `pingaccess-agent-apache22-rhel8*.zip`
- `apachectl` for the running IBM HTTP Server instance is in the file path.
- You've installed the Apache installation at `$IHS`. If you haven't, modify the file paths specified in this procedure based on where your Apache installation and configuration files are located.
- You've installed `libcurl` and PCRE or verified that they are installed. To install these packages, use the **`yum install libcurl pcre`** command.

Installing on a RHEL 8 system with an IBM HTTP Server

About this task

To manually install the PingAccess agent for Apache on RHEL 8 when using the IBM HTTP Server:

Steps

1. Go to the `pingaccess-agent-apache22-rhel8-<version>/<arch>/` directory.

   > **Note**
   >
   > Currently, the only valid value for *<arch>* is `x86_64` for 64-bit.

   Example:

   ```
   cd pingaccess-agent-apache22-rhel8-1.6.0/x86_64/
   ```

2. Extract the package RPMs using the following command:

   ```
   mkdir pkgroot
   cp *.rpm pkgroot/
   cd pkgroot
   for r in *.rpm; do rpm2cpio $r | cpio -idmv; done
   ```

3. Run the `cp` command to copy the libraries to the appropriate Apache directories.

   Example:

   For RedHat Enterprise Linux 8 (x86_64):

   ```
   cp -av usr/lib64/.so $IHS/modules
   ```

4. Copy `mod_paa.so` into the Apache modules directory:

   ```
   cp -av usr/lib64/httpd/modules/mod_paa.so $IHS/modules
   ```

5. Copy the `paa.conf` file to the Apache configuration directory:

   ```
   cp -av etc/httpd/conf.d/paa.conf $IHS/conf
   ```

6. In the `paa.conf` file:

   1. Add the following lines before the `LoadModule` directive:

      ```
      LoadFile modules/libpgm-5.2.so.0
      LoadFile modules/libzmq.so.5
      ```

   2. Update the values for `PaaPropertyFiles` and `PaaCertificateDir` to point to your Apache `conf` directory.

7. In the Apache configuration file, `$IHS/conf/httpd.conf`, use the following directive to add the PingAccess agent for Apache's module configuration:

```
Include conf/paa.conf
```

8. Copy the `<agentname>_agent.properties` file to the `$IHS/conf/agent.properties` directory.

   This is the configuration file that you downloaded in step 3 of Installing on an IBM HTTP Server using Apache 2.2.

9. Restart the Apache service by running the **`apachectl restart`** command.

Uninstalling the RHEL agent

Remove the PingAccess agent from a RHEL system.

About this task

- If you installed the PingAccess agent using the standard installation process, uninstall it with the following command:

```
sudo yum remove pingaccess-agent*.x86_64
```

- If you installed the PingAccess agent manually with an IBM HTTP Server, uninstall it with the following procedure:

Steps

1. Remove the `$APACHE/conf/agent.properties` file.

   Example:

   ```
   rm $APACHE/conf/agent.properties
   ```

2. Remove the following directive from the Apache configuration file, `$APACHE/conf/httpd.conf`.

   Example:

   ```
   Include conf/10-paa.conf
   ```

3. Remove `10-paa.conf` from the `$APACHE/modules` directory.

   Example:

   ```
   rm $APACHE/modules/10-paa.conf
   ```

4. Remove all `.so` files from the `$APACHE/modules` directory.

   Example:

   ```
   rm $APACHE/modules/.so
   ```

5. Restart the Apache service with the `apachectl restart` command.

RHEL agent configuration

Manage the PingAccess agent for RHEL configuration through the `paa.conf` and `agent.properties` configuration files.

The `/etc/httpd/conf.d/paa.conf` file contains the following 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 are `on` or `off`.
	This value can be set globally; set for individual virtual hosts, directories, locations, or files; or both. The agent follows the most specific value that you set.
	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 that store configuration data used to connect the agent to the PingAccess engine nodes that the agent will communicate with.	`conf.d/agent.properties`
`PaaEnabledNoteName`	An optional parameter that 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. To use this feature, you must deploy a custom module to include this note with the correct value.	`paa-enabled-note`

`Agent.properties`

The configured `agent.properties` files can contain the following properties:

Property	Definition	Default Value
`agent.engine.configuration.scheme`	The Uniform Resource Identifier (URI) scheme used to connect to the engine node. Acceptable values are: • `http` • `https`	`https`
`agent.engine.configuration.host`	The PingAccess host name.	The value in the agent node's `PingAccess Host` field.
`agent.engine.configuration.port`	The port that the agent connects to on the PingAccess host. **Tip** This value is defined in the PingAccess run.properties file.	Defined in the PingAccess admin console
`agent.engine.configuration.username`	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess admin console
`agent.engine.configuration.shared.secret`	The password which is used to authenticate the agent to the engine.	Defined in the PingAccess admin console

Property	Definition	Default Value
`agent.engine.configuration.bootstrap.truststore`	The base64-encoded public certificate which is 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, complete the following steps to verify that the Agent Trusted Certificate is configured correctly: 1. Base64 decode the public certificate into a `.crt` file and review the contents. 2. In the PingAccess server, make sure that the agent HTTP listener is using the matching private key. Learn more in Assigning key pairs.	Generated by PingAccess
`agent.engine.configuration.maxConnections`	The number of connections that a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host` property.	`10`
`agent.engine.configuration.timeout`	The maximum amount of time, in milliseconds, that an agent request made to PingAccess can take. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.engine.configuration.connectTimeout`	The maximum amount of time, in milliseconds, that the agent can take to connect to the PingAccess engine. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.cache.missInitialTimeout`	The maximum amount of time, in milliseconds, that 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 that 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`
`agent.cache.maxTokens`	The maximum number of tokens that are stored in the policy cache for a single web server worker process. A value of `0` means there is no maximum.	`0`

Property	Definition	Default Value
`agent.cache.disabled`	Determines whether policy decision caching 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. You might want to use this option 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. Because of the performance penalty, only use this option if necessary.	`0`
`agent.engine.configuration.failover.hosts`	The host name and port of the PingAccess server where the agent should send requests in the event of a failover from the PingAccess host.	Defined in the PingAccess admin console
`agent.engine.configuration.failover.failedRetryTimeout`	The number of seconds to wait before the agent should retry connecting to a failed PingAccess server.	`60`
`agent.engine.configuration.failover.MaxRetries`	The number of times to retry a connection to a PingAccess server after an unsuccessful attempt. If all retries fail, the agent marks the PingAccess server as failed for the duration of the `agent.engine.configuration.failover.failedRetryTimeout` value and tries another PingAccess server if one is available.	`2`
`agent.cache.type`	Controls the type of policy cache used by the agent. There are three acceptable values for this property: ***AUTO*** 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*** Does not share policy cache entries across worker processes. ***ZMQ*** Allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	`AUTO`

Property	Definition	Default Value
`agent.send.inventory`	Determines whether the `vnd-pi-agent` agent inventory header is sent along with each request to the PingAccess policy server. This header contains the following fields: **v** The PingAccess agent version. **t** The type of PingAccess agent retrieved using the `ap_get_server_description` function. **h** The host name of the PingAccess agent retrieved using the `ServerName` directive. Learn more in Agent inventory logging.	`true`
`agent.inventory`	Specifies additional values to include in the `vnd-pi-agent` agent inventory header. This property uses the following syntax: `agent.inventory=exampleheader=TEST;exampleheader2=TEST2;` **Note** The specified header fields are case sensitive.	This property isn't present by default.
`agent.apache.host.source.headerName`	If present, specifies a header that overrides the default `X-Forwarded-Host` header. This header communicates the authority component of the effective request Uniform Resource Locator (URL) on the protected application.	This property isn't present by default.
`agent.cache.defaultTokenType`	Specifies which token-type to favor when making an access decision if both a cookie and an authorization header token are included in a request. Acceptable values are `C` for cookie or `A` for authorization bearer token. Learn more in the **token-type**, **path**, and **vnd-pi-token-cache-oauth-ttl** entries in PAAP agent response in the PingAccess 8.1 documentation. **Note** This property isn't listed in the `agent.properties` file by default. To configure `A` as the `agent.cache.defaultTokenType`, you must add this property to the `agent.properties` file and set it equal to `A`.	`C`

Property	Definition	Default Value
`agent.request.block.xss.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request body. When defining these values, you can: • Use actual characters or URL-encoded characters • Specify a range of characters, such as a-z or `%00-%1f` • Use commas as delimiters to define multiple values **Note** To block a comma, you must URL encode it as `%2C`. • Configure any of the following special combinations for one value: ◦ Two forward slashes (//) ◦ A period and a forward slash (./) ◦ A forward slash and a period (/.) ◦ A forward slash and an asterisk (/*) ◦ An asterisk and a period (*.) The following example demonstrates how to block some common XSS characters: `agent.request.block.xss.characters=<,>,',/\,\,%22,%0a,%0d` **Note** Blocked requests are recorded as error entries in the PingAccess log. To get more details about why a particular request was blocked, set the log level to debug and review these error entries.	This property isn't present by default.
`agent.request.block.uri.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request URI. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common URI characters: `agent.request.block.uri.characters=//,./,/./,/,.,~,\,%00-%1f,%7f`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.query.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's query parameters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common query characters: `agent.request.block.query.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.form.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's form parameters. **Important** The request must have a `Content-Type` header value of `application/x-www-form-urlencoded` for the agent to block form characters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common form characters: `agent.request.block.form.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.xss.http.status`	Set a custom status code to display when the agent blocks a request because of a bad XSS character. **Tip** When configuring HTTP status codes initially, consider using a `500` error code to create more obvious test results. After you complete testing, set the HTTP status code to a more reasonable value, such as a `400` error code. The following example demonstrates how to set an XSS HTTP status code: `agent.request.block.xss.http.status=400`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.uri.http.status`	Set a custom status code to display when the agent blocks a request because of a bad URI character. The following example demonstrates how to set a URI HTTP status code: `agent.request.block.uri.http.status=404`	This property isn't present by default.
`agent.request.block.query.http.status`	Set a custom status code to display when the agent blocks a request because of a bad query character. The following example demonstrates how to set a query HTTP status code: `agent.request.block.query.http.status=400`	This property isn't present by default.
`agent.request.block.form.http.status`	Set a custom status code to display when the agent blocks a request because of a bad form character. The following example demonstrates how to set a form HTTP status code: `agent.request.block.form.http.status=400`	This property isn't present by default.

Tip

Add comments to the `agent.properties` files if necessary. The agent ignores lines beginning with the `#` or `!` characters.

Important

If you make changes to the `agent.properties` file, you must restart the web server.

Tip

Learn more about improving agent performance in the Performance tuning guide.

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.

Rotating a CA

Rotate the certificate authority (CA) used by an agent while minimizing the impact to agent communications.

Steps

1. On the agent web server, update the `agent.properties` file to add the new CA certificate.
    1. Concatenate the old and new CA certificates in PEM encoding format into a new file.
    2. Encode the contents of the file to Base64.
    3. Open the `agent.properties` file and set the value of the `agent.engine.configuration.bootstrap.truststore` line to the encoded content.

        Example:

        ```
        agent.engine.configuration.bootstrap.truststore=<Encoded_content>
        ```
2. Restart the agent web server.
3. Update the PingAccess configuration to use a new server certificate signed by the new CA for the agent HTTPS listener.
    1. Identify a key pair to use. If necessary, create a new key pair.

        Learn more in Generating new key pairs.
    2. Generate a CSR for that key pair.

        Learn more in Generating certificate signing requests.
    3. Submit that CSR to the new CA to get a new signed certificate.
    4. Import the CSR response (the new certificate) into PingAccess.

        Learn more in Importing certificates.
    5. Assign the key pair to the agent HTTPS listener.

        Learn more in Assigning key pairs to HTTPS listeners.

Troubleshooting

The following list indicates some potential problems and resolutions you might encounter with the PingAccess agent for RHEL.

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 SHA-256. If you cannot switch to SHA-256, you can re-enable SHA-1:

1. Open the `run.properties` file.
2. Add TLSv1 to the protocol list. For example:

```
tls.default.protocols=TLSv1, TLSv1.1, TLSv1.2, TLSv1.3
```

3. Add the SHA entries to the cipher suites list. For example:

```
tls.default.cipherSuites = TLS_CHACHA20_POLY1305_SHA256,\
                           TLS_AES_256_GCM_SHA384,\
                           TLS_AES_128_GCM_SHA256,\
                           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_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
```

PingAccess Agent for Apache (RHEL) Release Notes

The following release notes summarize the changes in current and previous PingAccess agent for Apache (RHEL) updates. Updated April 18, 2025.

> **Note**
>
> Removed RHEL 7 support.

PingAccess Agent for Apache (RHEL) 3.0 (April 2025)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 3.0.

Authenticate the PingAccess agent for Apache (RHEL) with a bearer token

New PASDKC-198

Authenticate PingAccess agents to the engine nodes with a stronger authentication method.

> **Note**
>
> To use this feature, you must upgrade to PingAccess 8.2 and the PingAccess agent for Apache (RHEL) 3.0 or later.

Learn more in the PingAccess 8.2 release notes. You can find setup instructions in Configuring PingAccess agents to use bearer token authentication and Agent SDK for C 3.0 (April 2025).

PingAccess Agent for Apache (RHEL) 1.6 (July 2024)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.4.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types. For more information, see the **Cache multiple token-types for Web + API applications** entry in the PingAccess 8.1 release notes, and the `agent.cache.defaultTokenType` property on the RHEL agent configuration page.

> **Note**
>
> Existing agent environments ignore the new `vnd-pi-token-cache-oauth-ttl` header and additional paths in the `vnd-pi-resource-cache` header.
> To see the performance boost, upgrade to PingAccess 8.1 and upgrade to the latest version of the RHEL agent. Otherwise, continue to use an earlier agent version.

Block bad characters in Apache and IIS agent deployments

New PAA-251

Configure a PingAccess Apache agent or the PingAccess agent for IIS to block requests that contain bad characters in the URI, query parameters, form parameters, or request body without having to reach out to PingAccess for a decision.

Added eight new properties to each agent:

1. `agent.request.block.xss.characters`
2. `agent.request.block.uri.characters`
3. `agent.request.block.query.characters`
4. `agent.request.block.form.characters`
5. `agent.request.block.xss.http.status`
6. `agent.request.block.uri.http.status`
7. `agent.request.block.query.http.status`
8. `agent.request.block.form.http.status`

Learn more in the configuration page for your agent:

- RHEL agent configuration
- SLES agent configuration

- Windows agent configuration
- IIS agent configuration

> **Note**
>
> For large scale or more complex blocking decisions, it's best practice for the agent to reach out to PingAccess for a decision.

PingAccess Agent for Apache (RHEL) 1.5.2 (September 2021)

Agent SDK for C version compatibility

Info

Compatible with Agent SDK for C version 1.3.

Override default `X-Forwarded-Host` header

New PAA-255

Added an option to override the default `X-Forwarded-Host` header with a specified header.

PingAccess Agent for Apache (RHEL) 1.5.1 (April 2021)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.3.

Fixed POST preservation corruption

Fixed PAA-220

Fixed an issue that caused large bodies sent through POST preservation in an agent deployment to be corrupted.

PingAccess Agent for Apache (RHEL) 1.5 (June 2020)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.3.

Removed support for RHEL 6

Info

Removed support for RHEL 6.

Added support for RHEL 8

New

Added support for RHEL 8.

Added ability to send agent inventory information to PingAccess

New PAA-178

Added agent inventory callback API.

PingAccess Agent for Apache (RHEL) 1.4.1 (February 2020)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.2.1.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for Apache (RHEL) 1.4 (June 2019)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.2.0.

Use `PAA Enabled` inside a directory or location container

Improved PAA-57

You can now use the `PAA Enabled` directive inside a directory or location container.

Set the policy caching mechanism in `agent.properties`

New PAA-75

Added ability to set policy caching mechanism using a property in the `agent.properties` file.

Manage agent processing for a request based on the note field

New PAA-90

Added ability to enable or disable agent processing for a request based on a note field.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for Apache (RHEL) 1.3.2 (November 2018)

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for Apache (RHEL) 1.3 (February 2017)

Added RHEL 6 support

New

Added support for Apache 2.4 on RHEL 6.

Disable the agent for specific hosts

New

The agent can be disabled for specific hosts using the new configuration option: `PaaEnabled`.

PingAccess Agent for Apache (RHEL) 1.2 (May 2016)

Added support for IBM HTTP Server

New

Added support for IBM HTTP Server.

PingAccess Agent for Apache (RHEL) 1.1 (December 2014)

Added RHEL 7 support

New

Added Support for Apache 2.4 on Red Hat Enterprise Linux 7.

Fixed a potential security issue

Security

Corrected a potential security issue related to caching, SECBL007. This security bulletin is available in the Ping Identity Support Portal.

PingAccess Agent for Apache (RHEL) 1.0 (July 2014)

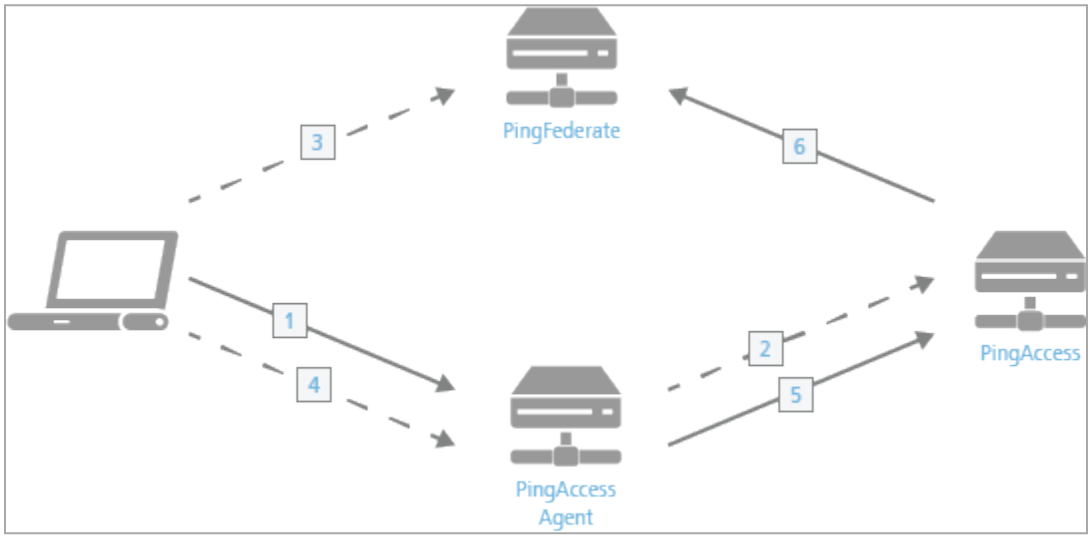
Initial release

Info

Initial release of the PingAccess Agent for Apache (RHEL).

PingAccess Agent for Apache (SLES)

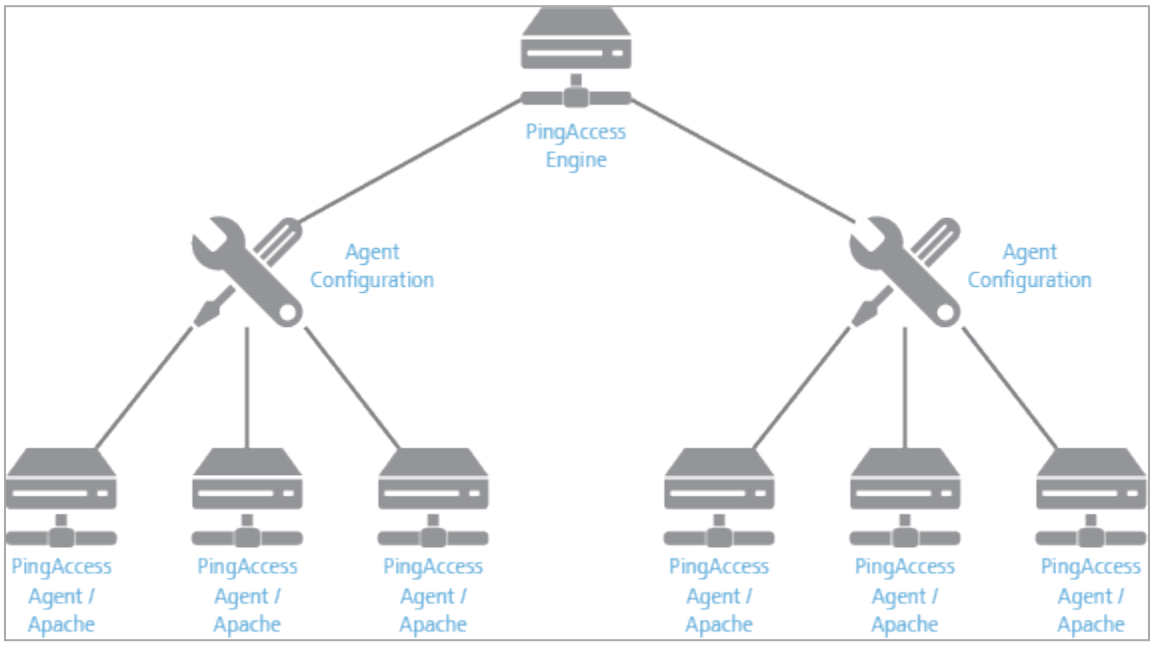
The PingAccess agent for Apache is an Apache module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



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 signs on, 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 administrative console, agent nodes are configured with information that allows an 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.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

Apache (SLES) agent system requirements

The PingAccess agent for Apache (SLES) is supported on the following platforms:

- Apache HTTP Server 2.4 running on SUSE Linux Enterprise Server 15 (x86_64). Use the latest supported SP from SUSE.

Note

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 you should deploy SELinux on the server.

Installing on SLES

Install a PingAccess agent on a SUSE Linux Enterprise Server (SLES) system.

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` file to modify the values for Apache's configuration and module directories.

Steps

1. Download the `pingaccess-agent-apache<version>.zip` archive from the **Add-ons** tab of the PingAccess downloads page and extract it.
2. Go to the `pingaccess-agent-apache<version>` directory.
    1. Import the gpg key.

        Example:

        ```
        rpm --import https://download.opensuse.org/repositories/network:/messaging:/zeromq:/release-stable/SLE_12_SP4/repodata/repomd.xml.key
        ```

    2. Install the dependencies.

        Example:

        ```
        zypper in ./x86_64/lib*.rpm
        ```

3. As root, copy the PingAccess agent for Apache files to the appropriate places.

    Example:

    ```
    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
    > ```

4. In the PingAccess console, go to **Applications > Agents**.
    1. Edit a configured agent.

        > **Note**
        >
        > If you haven't created an agent yet, learn more about how to do so in Adding agents.

2. In the **Shared Secrets** section, click the **Download** icon to download the configuration.

> **Note**
>
> The configuration file will be named `<agentname>_agent.properties`.

5. Copy the `<agentname>_agent.properties` file to `/etc/apache2/conf.d/agent.properties`.
6. As root, restart the Apache service using one of the following commands:

   Choose from:

   - `rcapache2 restart`
   - `$APACHE_ROOT/bin/apachectl restart`

Uninstalling on SLES

Remove the PingAccess agent from a SUSE Linux Enterprise Server (SLES) system.

Steps

- Remove the PingAccess agent for Apache files.

  Example:

```
rm /usr/lib64/apache2/mod_paa.so
rm /etc/apache2/conf.d/paa.conf
```

SLES agent configuration

Manage the PingAccess agent for SLES configuration through the `paa.conf` and `agent.properties` configuration files.

The `/etc/httpd/conf.d/paa.conf` file contains the following 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 are `on` or `off`.
	This value can be set globally; set for individual virtual hosts, directories, locations, or files; or both. The agent follows the most specific value that you set.
	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 that store configuration data used to connect the agent to the PingAccess engine nodes that the agent will communicate with.	`conf.d/agent.properties`
`PaaEnabledNoteName`	An optional parameter that 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. To use this feature, you must deploy a custom module to include this note with the correct value.	`paa-enabled-note`

> **Note**
>
> It isn't necessary to make any changes to `paa.conf` if you followed the steps in the Installation section.

`Agent.properties`

The configured `agent.properties` files can contain the following properties:

Property	Definition	Default Value
`agent.engine.configuration.scheme`	The Uniform Resource Identifier (URI) scheme used to connect to the engine node. Acceptable values are: • `http` • `https`	`https`
`agent.engine.configuration.host`	The PingAccess host name.	The value in the agent node's `PingAccess Host` field.
`agent.engine.configuration.port`	The port that the agent connects to on the PingAccess host. **Tip** This value is defined in the PingAccess run.properties file.	Defined in the PingAccess admin console
`agent.engine.configuration.username`	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess admin console
`agent.engine.configuration.shared.secret`	The password which is used to authenticate the agent to the engine.	Defined in the PingAccess admin console

Property	Definition	Default Value
`agent.engine.configuration.bootstrap.truststore`	The base64-encoded public certificate which is 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, complete the following steps to verify that the Agent Trusted Certificate is configured correctly: 1. Base64 decode the public certificate into a `.crt` file and review the contents. 2. In the PingAccess server, make sure that the agent HTTP listener is using the matching private key. Learn more in Assigning key pairs.	Generated by PingAccess
`agent.engine.configuration.maxConnections`	The number of connections that a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host` property.	`10`
`agent.engine.configuration.timeout`	The maximum amount of time, in milliseconds, that an agent request made to PingAccess can take. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.engine.configuration.connectTimeout`	The maximum amount of time, in milliseconds, that the agent can take to connect to the PingAccess engine. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.cache.missInitialTimeout`	The maximum amount of time, in milliseconds, that 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 that 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`
`agent.cache.maxTokens`	The maximum number of tokens that are stored in the policy cache for a single web server worker process. A value of `0` means there is no maximum.	`0`

Property	Definition	Default Value
`agent.cache.disabled`	Determines whether policy decision caching 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. You might want to use this option 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. Because of the performance penalty, only use this option if necessary.	`0`
`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.	Defined in the PingAccess admin console
`agent.engine.configuration.failover.failedRetryTimeout`	The number of seconds to wait before the agent should retry connecting to a failed PingAccess server.	`60`
`agent.engine.configuration.failover.MaxRetries`	The number of times to retry a connection to a PingAccess server after an unsuccessful attempt. If all retries fail, the agent marks the PingAccess server as failed for the duration of the `agent.engine.configuration.failover.failedRetryTimeout` value and tries another PingAccess server if one is available.	`2`
`agent.cache.type`	Controls the type of policy cache used by the agent. There are three acceptable values for this property: ***AUTO*** 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*** Does not share policy cache entries across worker processes. ***ZMQ*** Allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	`AUTO`

Property	Definition	Default Value
`agent.send.inventory`	Determines whether the `vnd-pi-agent` agent inventory header is sent along with each request to the PingAccess policy server. This header contains the following fields: **v** The PingAccess agent version. **t** The type of PingAccess agent retrieved using the `ap_get_server_description` function. **h** The hostname of the PingAccess agent retrieved using the `ServerName` directive. Learn more in Agent inventory logging.	`true`
`agent.inventory`	Specifies additional values to include in the `vnd-pi-agent` agent inventory header. This property uses the following syntax: `agent.inventory=exampleheader=TEST;exampleheader2=TEST2;` **Note** The specified header fields are case-sensitive.	This property isn't present by default.
`agent.apache.host.source.headerName`	If present, specifies a header that overrides the default `X-Forwarded-Host` header. This header communicates the authority component of the effective request Uniform Resource Locator (URL) on the protected application.	This property isn't present by default.
`agent.cache.defaultTokenType`	Specifies which token-type to favor when making an access decision if both a cookie and an authorization header token are included in a request. Acceptable values are `C` for cookie or `A` for authorization bearer token. Learn more in the **token-type**, **path**, and **vnd-pi-token-cache-oauth-ttl** entries in PAAP agent response in the PingAccess 8.1 documentation. **Note** This property isn't listed in the `agent.properties` file by default. To configure `A` as the `agent.cache.defaultTokenType`, you must add this property to the `agent.properties` file and set it equal to `A`.	`C`

Property	Definition	Default Value
`agent.request.block.xss.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request body. When defining these values, you can: • Use actual characters or URL-encoded characters • Specify a range of characters, such as a-z or `%00-%1f` • Use commas as delimiters to define multiple values **Note** To block a comma, you must URL encode it as `%2C`. • Configure any of the following special combinations for one value: ◦ Two forward slashes (//) ◦ A period and a forward slash (./) ◦ A forward slash and a period (/.) ◦ A forward slash and an asterisk (/*) ◦ An asterisk and a period (*.) The following example demonstrates how to block some common XSS characters: `agent.request.block.xss.characters=<,>,',/\,\,%22,%0a,%0d` **Note** Blocked requests are recorded as error entries in the PingAccess log. To get more details about why a particular request was blocked, set the log level to debug and review these error entries.	This property isn't present by default.
`agent.request.block.uri.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request URI. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common URI characters: `agent.request.block.uri.characters=//,./,/.,/,.,~,\,%00-%1f,%7f`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.query.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's query parameters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common query characters: `agent.request.block.query.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.form.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's form parameters. **Important** The request must have a `Content-Type` header value of `application/x-www-form-urlencoded` for the agent to block form characters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common form characters: `agent.request.block.form.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.xss.http.status`	Set a custom status code to display when the agent blocks a request because of a bad XSS character. **Tip** When configuring HTTP status codes initially, consider using a `500` error code to create more obvious test results. After you complete testing, set the HTTP status code to a more reasonable value, such as a `400` error code. The following example demonstrates how to set an XSS HTTP status code: `agent.request.block.xss.http.status=400`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.uri.http.status`	Set a custom status code to display when the agent blocks a request because of a bad URI character. The following example demonstrates how to set a URI HTTP status code: `agent.request.block.uri.http.status=404`	This property isn't present by default.
`agent.request.block.query.http.status`	Set a custom status code to display when the agent blocks a request because of a bad query character. The following example demonstrates how to set a query HTTP status code: `agent.request.block.query.http.status=400`	This property isn't present by default.
`agent.request.block.form.http.status`	Set a custom status code to display when the agent blocks a request because of a bad form character. The following example demonstrates how to set a form HTTP status code: `agent.request.block.form.http.status=400`	This property isn't present by default.

Tip

Add comments to the `agent.properties` files if necessary. The agent ignores lines beginning with the `#` or `!` characters.

Important

If you make changes to the `agent.properties` file, you must restart the web server.

Tip

Learn more about improving agent performance in the Performance tuning guide.

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 agent is prefaced with the string `[paa]`. Agent monitoring and performance information is prefaced with the string `[paa-monitoring]`, and contains information about how long the agent took to fill a cache request and how long the total policy decision took.

The LogLevel used by the agent module is taken from the top-level `httpd.conf` configuration.

Rotating a CA

Rotate the certificate authority (CA) used by an agent while minimizing the impact to agent communications.

Steps

1. On the agent web server, update the `agent.properties` file to add the new CA certificate.
    1. Concatenate the old and new CA certificates in PEM encoding format into a new file.
    2. Encode the contents of the file to Base64.
    3. Open the `agent.properties` file and set the value of the `agent.engine.configuration.bootstrap.truststore` line to the encoded content.

        Example:

        ```
        agent.engine.configuration.bootstrap.truststore=<Encoded_content>
        ```
2. Restart the agent web server.
3. Update the PingAccess configuration to use a new server certificate signed by the new CA for the agent HTTPS listener.
    1. Identify a key pair to use. If necessary, create a new key pair.

        Learn more in Generating new key pairs.
    2. Generate a CSR for that key pair.

        Learn more in Generating certificate signing requests.
    3. Submit that CSR to the new CA to get a new signed certificate.
    4. Import the CSR response (the new certificate) into PingAccess.

        Learn more in Importing certificates.
    5. Assign the key pair to the agent HTTPS listener.

        Learn more in Assigning key pairs to HTTPS listeners.

Troubleshooting

The following table lists some potential problems and resolutions you might encounter with the PingAccess agent for SUSE Linux Enterprise Server (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 SHA-256. If you cannot switch to SHA-256, you can re-enable SHA-1:
	1. Open the `run.properties` file.
	2. Add TLSv1 to the protocol list.
	`tls.default.protocols=TLSv1, TLSv1.1, TLSv1.2, TLSv1.3`
	3. Add the SHA entries to the cipher suites list.
	`tls.default.cipherSuites = TLS_CHACHA20_POLY1305_SHA256,\ TLS_AES_256_GCM_SHA384,\ TLS_AES_128_GCM_SHA256,\ 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_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`

Agent for Apache (SLES) Release Notes

These release notes summarize the changes in current and previous PingAccess agent for Apache (SLES) updates. Updated April 18, 2025.

> **Note**
>
> Removed SLES 12 support.

PingAccess Agent for Apache (SLES) 3.0 (April 2025)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 3.0.

Authenticate the PingAccess agent for Apache (SLES) with a bearer token

New PASDKC-198

Authenticate PingAccess agents to the engine nodes with a stronger authentication method.

> **Note**
>
> To use this feature, you must upgrade to PingAccess 8.2 and the PingAccess agent for Apache (SLES) 3.0 or later.

Learn more in the PingAccess 8.2 release notes. You can find setup instructions in Configuring PingAccess agents to use bearer token authentication and Agent SDK for C 3.0 (April 2025).

PingAccess Agent for Apache (SLES) 1.6 (July 2024)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.4.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types. For more information, see the **Cache multiple token-types for Web + API applications** entry in the PingAccess 8.1 release notes, and the `agent.cache.defaultTokenType` property on the SLES agent configuration page.

> **Note**
>
> Existing agent environments ignore the new `vnd-pi-token-cache-oauth-ttl` header and additional paths in the `vnd-pi-resource-cache` header.
> To see the performance boost, upgrade to PingAccess 8.1 and upgrade to the latest version of the SLES agent. Otherwise, continue to use an earlier agent version.

Block bad characters in Apache and IIS agent deployments

New PAA-251

Configure a PingAccess Apache agent or the PingAccess agent for IIS to block requests that contain bad characters in the URI, query parameters, form parameters, or request body without having to reach out to PingAccess for a decision.

Added eight new properties to each agent:

1. `agent.request.block.xss.characters`
2. `agent.request.block.uri.characters`
3. `agent.request.block.query.characters`

4. `agent.request.block.form.characters`
5. `agent.request.block.xss.http.status`
6. `agent.request.block.uri.http.status`
7. `agent.request.block.query.http.status`
8. `agent.request.block.form.http.status`

Learn more in the configuration page for your agent:

- RHEL agent configuration
- SLES agent configuration
- Windows agent configuration
- IIS agent configuration

> **Note**
>
> For large scale or more complex blocking decisions, it's best practice for the agent to reach out to PingAccess for a decision.

PingAccess Agent for Apache (SLES) 1.5.2 (September 2021)

Agent SDK for C version compatibility

Info

Compatible with Agent SDK for C version 1.3.

Override default `X-Forwarded-Host` header

New PAA-255

Added an option to override the default `X-Forwarded-Host` header with a specified header.

PingAccess Agent for Apache (SLES) 1.5.1 (April 2021)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.3.

Fixed POST preservation corruption

Fixed PAA-220

Fixed an issue that caused large bodies sent through POST preservation in an agent deployment to be corrupted.

PingAccess Agent for Apache (SLES) 1.5 (July 2020)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.3.

Added ability to send agent inventory information toPingAccess

New PAA-178

Added agent inventory callback API.

PingAccess Agent for Apache (SLES) 1.4.1 (February 2020)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.2.1.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for Apache (SLES) 1.4 (June 2019)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.2.0.

Use PAA Enabled inside a directory or location container

Improved PAA-57

You can now use the **PAA Enabled** directive inside a directory or location container.

Set the policy caching mechanism in `agent.properties`

New PAA-75

Added ability to set policy caching mechanism using a property in the `agent.properties` file.

Manage agent processing for a request based on the note field

New PAA-90

Added ability to enable or disable agent processing for a request based on a note field.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for Apache (SLES) 1.3.2 (November 2018)

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for Apache (SLES) 1.3 (March 2017)

Initial release

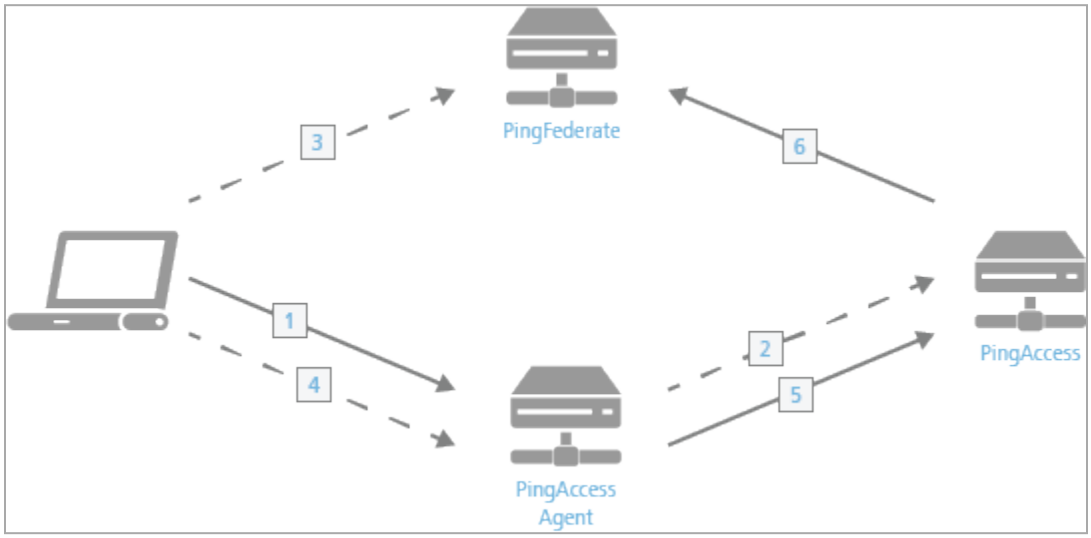
Info

Initial release for Apache 2.2 on SUSE Linux Enterprise Server (SLES) 11 and Apache 2.4 on SLES 12.

> **Note**
>
> Version is aligned with PingAccess agent for Apache (RHEL).

PingAccess Agent for Apache (Windows)

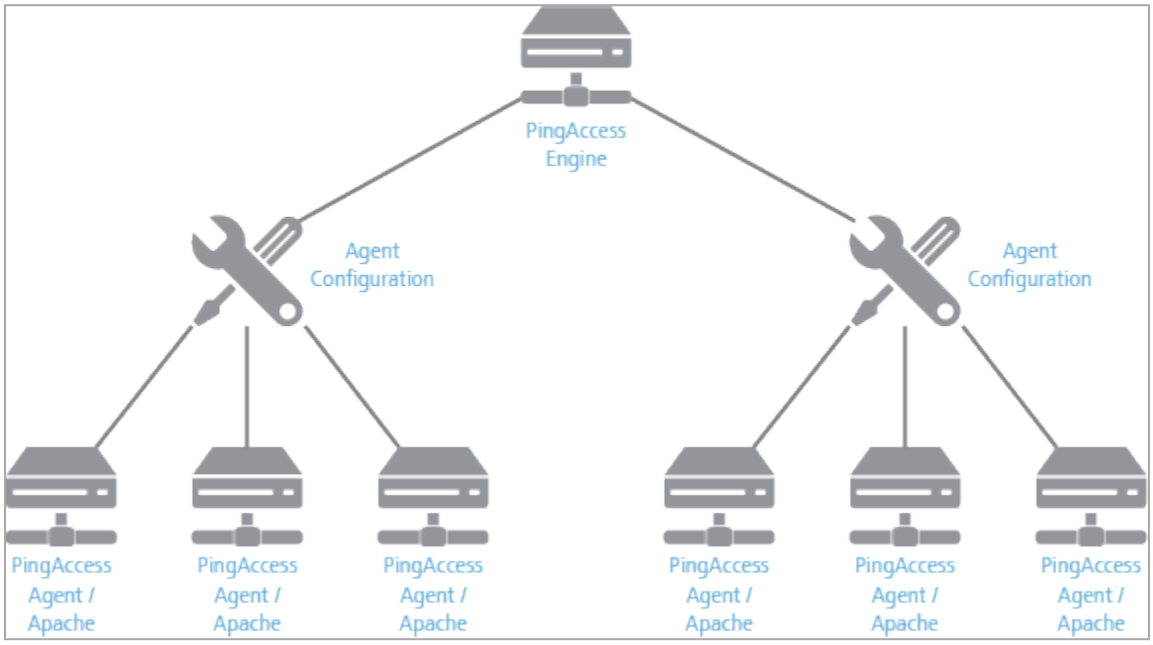
The PingAccess agent for Apache is an Apache module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



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 signs on, 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 administrative 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.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

Apache (Windows) agent system requirements

The PingAccess agent for Apache (Windows) is supported on these platforms.

- Apache HTTP Server 2.4 64-bit running on Microsoft Windows Server 2016, VC14 or later
- Apache HTTP Server 2.4 64-bit running on Microsoft Windows Server 2019, VC14 or later
- Apache HTTP Server 2.4 64-bit running on Microsoft Windows Server 2022, VC14 or later

As with any system that is reachable from the internet, the server should be properly hardened.

Installing on Windows

Install a PingAccess agent on a Windows system.

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.
- To modify the values for Apache's configuration and module directories, edit the included `paa.conf` file.

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 **Applications > Agents**.
7. Edit a configured agent.

   If the agent has not yet been created, see Adding agents.
8. In the shared secret, click **Download** to download the configuration.

   The configuration file is 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.

Uninstalling on Windows

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.

Windows agent configuration

Manage the PingAccess agent for Windows configuration through the `paa.conf` and `agent.properties` configuration files.

The `C:/Apache24/conf/paa.conf` file contains the following 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 are `on` or `off`.
	This value can be set globally; set for individual virtual hosts, directories, locations, or files; or both. The agent follows the most specific value that you set.
	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 /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 that store configuration data used to connect the agent to the PingAccess engine nodes that the agent will communicate with.	`conf.d/agent.properties`
`PaaEnabledNoteName`	An optional parameter that 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. To use this feature, you must deploy a custom module to include this note with the correct value.	`paa-enabled-note`

`Agent.properties`

The configured `agent.properties` files can contain the following properties:

Property	Definition	Default Value
`agent.engine.configuration.scheme`	The Uniform Resource Identifier (URI) scheme used to connect to the engine node. Acceptable values are: • `http` • `https`	`https`
`agent.engine.configuration.host`	The PingAccess host name.	The value in the agent node's `PingAccess Host` field.
`agent.engine.configuration.port`	The port that the agent connects to on the PingAccess host. **Tip** This value is defined in the PingAccess run.properties file.	Defined in the PingAccess admin console
`agent.engine.configuration.username`	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess admin console
`agent.engine.configuration.shared.secret`	The password which is used to authenticate the agent to the engine.	Defined in the PingAccess admin console

Property	Definition	Default Value
`agent.engine.configuration.bootstrap.truststore`	The base64-encoded public certificate which is 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, complete the following steps to verify that the Agent Trusted Certificate is configured correctly: 1. Base64 decode the public certificate into a `.crt` file and review the contents. 2. In the PingAccess server, make sure that the agent HTTP listener is using the matching private key. Learn more in Assigning key pairs.	Generated by PingAccess
`agent.engine.configuration.maxConnections`	The number of connections that a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host` property.	`10`
`agent.engine.configuration.timeout`	The maximum amount of time, in milliseconds, that an agent request made to PingAccess can take. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.engine.configuration.connectTimeout`	The maximum amount of time, in milliseconds, that the agent can take to connect to the PingAccess engine. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.cache.missInitialTimeout`	The maximum amount of time, in milliseconds, that 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 that 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
`agent.cache.maxTokens`	The maximum number of tokens that are stored in the policy cache for a single web server worker process. A value of `0` means there is no maximum.	0

Property	Definition	Default Value
`agent.cache.disabled`	Determines whether policy decision caching 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. You might want to use this option 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. Because of the performance penalty, only use this option if necessary.	0
`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.	Defined in the PingAccess admin console
`agent.engine.configuration.failover.failedRetryTimeout`	The number of seconds to wait before the agent should retry connecting to a failed PingAccess server.	`60`
`agent.engine.configuration.failover.MaxRetries`	The number of times to retry a connection to a PingAccess server after an unsuccessful attempt. If all retries fail, the agent marks the PingAccess server as failed for the duration of the `agent.engine.configuration.failover.failedRetryTimeout` value and tries another PingAccess server if one is available.	`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, 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`

Property	Definition	Default Value
`agent.send.inventory`	Determines whether the `vnd-pi-agent` agent inventory header is sent along with each request to the PingAccess policy server. This header contains the following fields: **v** The PingAccess agent version. **t** The type of PingAccess agent retrieved using the `ap_get_server_description` function. **h** The hostname of the PingAccess agent retrieved using the `ServerName` directive. Learn more in Agent inventory logging.	`true`
`agent.inventory`	Specifies additional values to include in the `vnd-pi-agent` agent inventory header. This property uses the following syntax: `agent.inventory=exampleheader=TEST;exampleheader2=TEST2;` **Note** The specified header fields are case-sensitive.	This property isn't present by default.
`agent.apache.host.source.headerName`	If present, specifies a header that overrides the default `X-Forwarded-Host` header. This header communicates the authority component of the effective request Uniform Resource Locator (URL) on the protected application.	This property isn't present by default.
`agent.cache.defaultTokenType`	Specifies which token-type to favor when making an access decision if both a cookie and an authorization header token are included in a request. Acceptable values are `C` for cookie or `A` for authorization bearer token. Learn more in the **token-type**, **path**, and **vnd-pi-token-cache-oauth-ttl** entries in PAAP agent response in the PingAccess 8.1 documentation. **Note** This property isn't listed in the `agent.properties` file by default. To configure `A` as the `agent.cache.defaultTokenType`, you must add this property to the `agent.properties` file and set it equal to `A`.	`C`

Property	Definition	Default Value
`agent.request.block.xss.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request body. When defining these values, you can: • Use actual characters or URL-encoded characters • Specify a range of characters, such as a-z or `%00-%1f` • Use commas as delimiters to define multiple values **Note** To block a comma, you must URL encode it as `%2C`. • Configure any of the following special combinations for one value: ◦ Two forward slashes (//) ◦ A period and a forward slash (./) ◦ A forward slash and a period (/.) ◦ A forward slash and an asterisk (/*) ◦ An asterisk and a period (*.) The following example demonstrates how to block some common XSS characters: `agent.request.block.xss.characters=<,>,',/\,\,%22,%0a,%0d` **Note** Blocked requests are recorded as error entries in the PingAccess log. To get more details about why a particular request was blocked, set the log level to debug and review these error entries.	This property isn't present by default.
`agent.request.block.uri.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request URI. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common URI characters: `agent.request.block.uri.characters=//,./,/.,/,.,~,\,%00-%1f,%7f`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.query.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's query parameters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common query characters: `agent.request.block.query.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.form.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's form parameters. **Important** The request must have a `Content-Type` header value of `application/x-www-form-urlencoded` for the agent to block form characters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common form characters: `agent.request.block.form.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.xss.http.status`	Set a custom status code to display when the agent blocks a request because of a bad XSS character. **Tip** When configuring HTTP status codes initially, consider using a `500` error code to create more obvious test results. After you complete testing, set the HTTP status code to a more reasonable value, such as a `400` error code. The following example demonstrates how to set an XSS HTTP status code: `agent.request.block.xss.http.status=400`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.uri.http.status`	Set a custom status code to display when the agent blocks a request because of a bad URI character. The following example demonstrates how to set a URI HTTP status code: `agent.request.block.uri.http.status=404`	This property isn't present by default.
`agent.request.block.query.http.status`	Set a custom status code to display when the agent blocks a request because of a bad query character. The following example demonstrates how to set a query HTTP status code: `agent.request.block.query.http.status=400`	This property isn't present by default.
`agent.request.block.form.http.status`	Set a custom status code to display when the agent blocks a request because of a bad form character. The following example demonstrates how to set a form HTTP status code: `agent.request.block.form.http.status=400`	This property isn't present by default.

Tip

Add comments to the `agent.properties` files if necessary. The agent ignores lines beginning with the `#` or `!` characters.

Important

If you make changes to the `agent.properties` file, you must restart the web server.

Tip

Learn more about improving agent performance in the Performance tuning guide.

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 agent is prefaced with the string `[paa]`. Agent monitoring and performance information is prefaced with the string `[paa-monitoring]` and contains information about how long the 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.

Rotating a CA

Rotate the certificate authority (CA) used by an agent while minimizing the impact to agent communications.

Steps

1. On the agent web server, update the `agent.properties` file to add the new CA certificate.
    1. Concatenate the old and new CA certificates in PEM encoding format into a new file.
    2. Encode the contents of the file to Base64.
    3. Open the `agent.properties` file and set the value of the `agent.engine.configuration.bootstrap.truststore` line to the encoded content.

        Example:

        ```
        agent.engine.configuration.bootstrap.truststore=<Encoded_content>
        ```
2. Restart the agent web server.
3. Update the PingAccess configuration to use a new server certificate signed by the new CA for the agent HTTPS listener.
    1. Identify a key pair to use. If necessary, create a new key pair.

        Learn more in Generating new key pairs.
    2. Generate a CSR for that key pair.

        Learn more in Generating certificate signing requests.
    3. Submit that CSR to the new CA to get a new signed certificate.
    4. Import the CSR response (the new certificate) into PingAccess.

        Learn more in Importing certificates.
    5. Assign the key pair to the agent HTTPS listener.

        Learn more in Assigning key pairs to HTTPS listeners.

PingAccess Agent for Apache (Windows) release notes

These release notes summarize the changes in current and previous PingAccess Agent for Apache (Windows) updates. Updated July 18, 2024.

PingAccess Agent for Apache (Windows) 1.6 (July 2024)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.4.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types. For more information, see the **Cache multiple token-types for Web + API applications** entry in the PingAccess 8.1 release notes, and the `agent.cache.defaultTokenType` property on the Windows agent configuration page.

> **Note**
>
> Existing agent environments ignore the new `vnd-pi-token-cache-oauth-ttl` header and additional paths in the `vnd-pi-resource-cache` header.
> To see the performance boost, upgrade to PingAccess 8.1 and upgrade to the latest version of the Windows agent. Otherwise, continue to use an earlier agent version.

Block bad characters in Apache and IIS agent deployments

New PAA-251

Configure a PingAccess Apache agent or the PingAccess agent for IIS to block requests that contain bad characters in the URI, query parameters, form parameters, or request body without having to reach out to PingAccess for a decision.

Added eight new properties to each agent:

1. `agent.request.block.xss.characters`
2. `agent.request.block.uri.characters`
3. `agent.request.block.query.characters`
4. `agent.request.block.form.characters`
5. `agent.request.block.xss.http.status`
6. `agent.request.block.uri.http.status`
7. `agent.request.block.query.http.status`
8. `agent.request.block.form.http.status`

Learn more in the configuration page for your agent:

- RHEL agent configuration
- SLES agent configuration
- Windows agent configuration
- IIS agent configuration

> **Note**
>
> For large scale or more complex blocking decisions, it's best practice for the agent to reach out to PingAccess for a decision.

PingAccess Agent for Apache (Windows) 1.5.2 (September 2021)

Agent SDK for C version compatibility

Info

Compatible with Agent SDK for C version 1.3.

Override default `X-Forwarded-Host` header

New PAA-255

Added an option to override the default `X-Forwarded-Host` header with a specified header.

PingAccess Agent for Apache (Windows) 1.5.1 (April 2021)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.3.

Fixed POST preservation corruption

Fixed PAA-220

Fixed an issue that caused large bodies sent through POST preservation in an agent deployment to be corrupted.

PingAccess Agent for Apache (Windows) 1.5 (July 2020)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.3.

Added ability to send agent inventory information toPingAccess

New PAA-178

Added agent inventory callback API.

PingAccess Agent for Apache (Windows) 1.4.1 (February 2020)

Agent SDK for C compatibility

Info

Compatible with Agent SDK for C version 1.2.1.

Fixed a potential security issue

Security

Fixed a potential security issue

PingAccess Agent for Apache (Windows) 1.4 (July 2019)

Agent SDK for C version compatibility

Info

Compatible with Agent SDK for C version 1.2.0.

Initial release

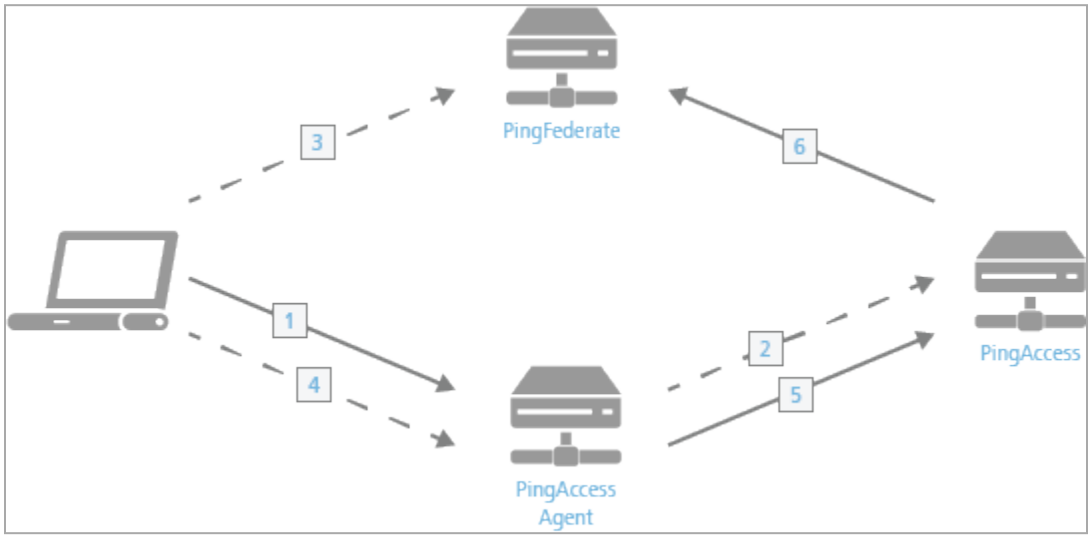
Info

Initial release for Apache 2.4 on Windows.

> **Note**
>
> Version is aligned with the PingAccess Agent for Apache (RHEL).

PingAccess Agent for IIS

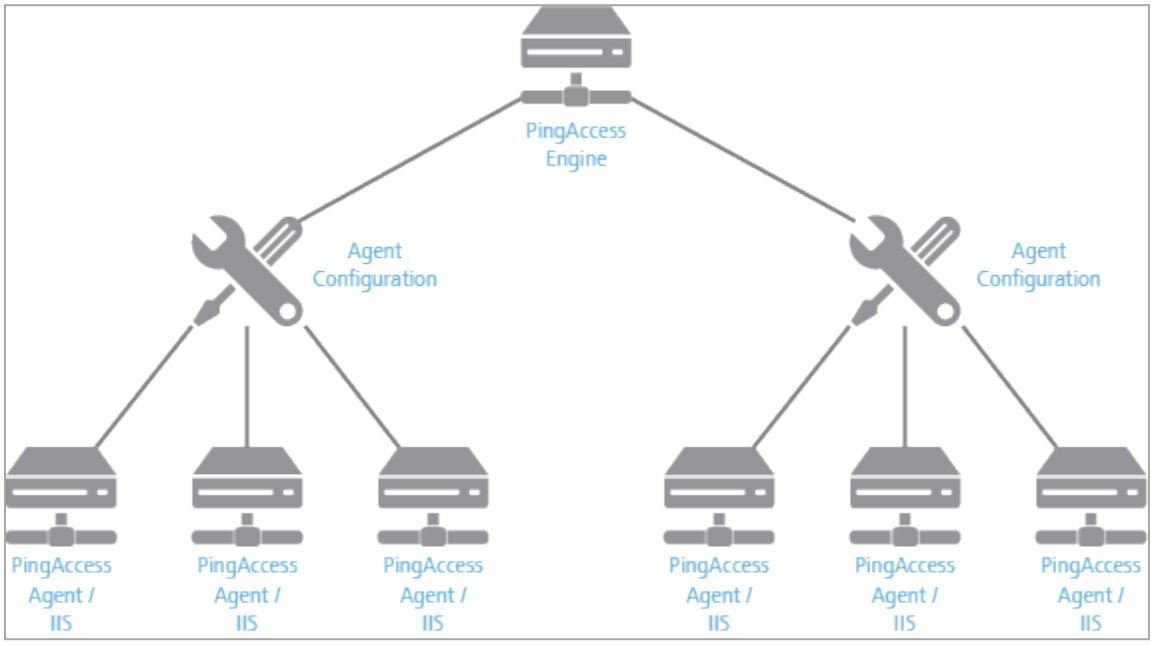
The PingAccess agent for IIS is a Microsoft Internet Information Services module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



Processing steps

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 administrative 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

A problem with the PingAccess IIS agent configuration might 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.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

IIS agent system requirements

The PingAccess agent for IIS is supported on these platforms.

- Microsoft Internet Information Services (IIS) 10.0 running on Windows Server 2016
- Microsoft Internet Information Services (IIS) 10.0 running on Windows Server 2019
- Microsoft Internet Information Services (IIS) 10.0 running on Windows Server 2022

Note

Only 64-bit Windows platforms and versions of IIS are supported. 32-bit compatibility mode for IIS is not supported by this agent.

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.

Installing on IIS

Install a PingAccess agent on an Internet Information Services (IIS) system.

Before you begin

Shut down any running programs, including the Windows Event Viewer. Running applications might 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.

Steps

1. Extract the PingAccess agent for IIS `.zip` file.

   > **Note**
   >
   > The installer cannot be run from inside the `.zip` file. You must first extract it.

2. Run the `pingaccess-agent-iis.msi` installer.
   1. Click **Next**.
   2. **Optional:** Specify a new destination folder, and then click **Next**.
   3. Click **Install**.
   4. Click **Finish**.
3. In the PingAccess console, go to **Applications > Agents**.
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 is named `<agentname>_agent.properties`.
6. To create the `C:\Program Files\Ping Identity\PingAccess Agent for IIS\agent.properties` file on the agent system, copy and rename the `<agentname>_agent.properties` file. .

   > **Note**
   >
   > If you changed the destination folder for the agent, update the file path for the `agent.properties` file to match the new location.

7. Restart Microsoft IIS Server on the system:
   1. Launch IIS Manager.
   2. Navigate to the web server node in the Connections tree.
   3. In the **Actions** pane, click **Restart**.

> **Tip**
>
> You can restart the IIS service by running the command `iisreset /restart`.

Manually Installing on IIS

Manually install a PingAccess agent for Internet Information Services (IIS), or if the installation failed, manually complete a partial installation.

About this task

> **Important**
>
> For information about preventing a known issue on systems running application pools in 32-bit compatibility mode, see Troubleshooting.

> **Tip**
>
> If you use this procedure due to an installation problem, open a support ticket so the underlying issue can be addressed.

Steps

1. Stop Microsoft IIS:
    1. Run the command `net stop w3svc`.
    2. Run the command `net stop was`.
2. Extract the `pingaccess-agent-iis.msi` installer file from the PingAccess IIS Agent Distribution `pingaccess-agent-iis-x.x.x.zip` file.
3. Extract the MSI installer file's contents.

    ```
    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:
    1. `cd C:\<TARGETDIR>\PFiles\inetsrv\config\schema\`
    2. `copy paa_schema.xml C:\Windows\System32\inetsrv\config\schema\`

7. Edit `C:\Windows\System32\inetsrv\config\applicationHost.config` and make the following changes:

    1. Add `sectionGroup` to the container with `name=system.webServer` under `configSections`.

        Example:

        ```
        <section name="paa" overrideModeDefault="Deny" allowDefinition="AppHostOnly"
        allowLocation="false" />
        ```

    2. Add the following XML block to the `<system.webServer>` element.

        Example:

        ```
        <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>
        ```

8. Open IIS Manager and go to **Management → Configuration Editor**.

9. Select the `system.webServer/paa` section and validate that the paths added to `applicationHost.config` have the following values:

    ### *paaCertificateDir*

    `C:\Program Files\Ping Identity\PingAccess Agent for IIS\certs\`

    ### *paaPropertyFiles*

    (Count=1)

    > **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. Verify that the `C:\Program Files\Ping Identity\PingAccess Agent for IIS\certs` folder has been created.

11. Change the permissions of `C:\Program Files\Ping Identity\PingAccess Agent for IIS\certs` to include read and write permissions for `IIS_IUSRS`.

    You might need to manually search for this user when modifying the permissions.

12. Register the PingAccess agent logging publisher:

    1. Run the following 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"
        ```

2. 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:
    1. Open IIS Manager, then select the web server the agent is being added to.
    2. Click **Modules**.
    3. Click **Configure Native Modules**.
    4. Click **Register** and enter the following information.

Name	`PingAccessAgentModule`
Path	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\paa-iis-module.dll`

    5. Click **OK**.
    6. Click **OK**.
    7. 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.

Result

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.

Uninstalling on IIS

Remove the PingAccess agent from an Internet Information Services (IIS) system.

About this task

> **Note**
>
> Alternatively, 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.

   Result:

   The installer displays the available workflows.

2. Select the **Remove** workflow.

IIS agent configuration

Manage the PingAccess agent for Internet Information Services (IIS) configuration using the 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 following two configuration options:

Parameter	Definition	Default Value
`PaaCertificateDir`	A string value containing the path to the certificates extracted from the `.properties` files.	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\certs.properties`
`PaaPropertyFiles`	The list of `.properties` files which store configuration data used to connect the agent to the PingAccess engine nodes with which the agent will communicate.	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\agent.properties`

> **Important**
>
> Do not make any changes to the previous two configuration options if you followed the steps in Installing on IIS.

`Agent.properties`

The configured `agent.properties` files can contain the following properties:

Property	Definition	Default Value
`agent.engine.configuration.scheme`	The Uniform Resource Identifier (URI) scheme used to connect to the engine node. Acceptable values are: • `http` • `https`	`https`
`agent.engine.configuration.host`	The PingAccess host name.	The value in the agent node's **PingAccess Host** field.

Property	Definition	Default Value
`agent.engine.configuration.port`	The port that the agent connects to on the PingAccess host. **Tip** 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 (CRL) 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.	This property isn't present by default. The value is treated as `1` when not specified.
`agent.engine.configuration.checkCertRevocation.bestEffort`	Determines whether the agent ignores the requirement to perform CRL checking if the backend server is offline or missing. A value of `1` enables the ability to ignore CRL checking, while a value of `0` disables the property. **Note** If the value of the `agent.engine.configuration.checkCertRevocation` property is `0`, the agent ignores the `agent.engine.configuration.checkCertRevocation.bestEffort` property as well.	This property isn't present by default. The value is treated as `0` when not specified.
`agent.engine.configuration.shared.secret`	The password which is used to authenticate the agent to the engine.	Defined in the PingAccess admin UI.
`agent.engine.configuration.bootstrap.truststore`	The base64-encoded public certificate which is 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, complete the following steps to verify that the Agent Trusted Certificate is configured correctly: 1. Base64 decode the public certificate into a `.crt` file and review the contents. 2. In the PingAccess server, make sure that the agent HTTP listener is using the matching private key. Learn more in Assigning key pairs.	Generated by PingAccess.
`agent.engine.configuration.maxConnections`	The number of connections that a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host` property.	`10`

Property	Definition	Default Value
`agent.engine.configuration.timeout`	The maximum amount of time, in milliseconds, that an agent request made to PingAccess can take. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.engine.configuration.connectTimeout`	The maximum amount of time, in milliseconds, that the agent can take to connect to the PingAccess engine. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.cache.missInitialTimeout`	The maximum amount of time, in milliseconds, that 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 that 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`
`agent.cache.maxTokens`	The maximum number of tokens that are 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 policy decision caching 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. You might want to use this option 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. Because of the performance penalty, only use this option if necessary.	`0`
`agent.engine.configuration.failover.hosts`	The host name and port of the PingAccess server where the agent should send requests in the event of a failover from the PingAccess host.	Defined in the PingAccess admin console
`agent.engine.configuration.failover.failedRetryTimeout`	The number of seconds to wait before the agent should retry connecting to a failed PingAccess server.	`60`

Property	Definition	Default Value
`agent.engine.configuration.failover.MaxRetries`	The number of times to retry a connection to a PingAccess server after an unsuccessful attempt. If all retries fail, the agent marks the PingAccess server as failed for the duration of the `agent.engine.configuration.failover.failedRetryTimeout` value and tries another PingAccess server if one is available.	`2`
`agent.cache.type`	Controls the type of policy cache used by the agent. There are three acceptable values for this property: ***AUTO*** 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*** Does not share policy cache entries across worker processes. ***ZMQ*** Allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	`AUTO`
`agent.send.inventory`	Determines whether the `vnd-pi-agent` agent inventory header is sent along with each request to the PingAccess policy server. This header contains the following fields: **v** The PingAccess agent version. **t** The type of PingAccess agent retrieved from the server variable on the IIS context, which returns a string such as `Microsoft-IIS/7.5`. **h** The host name of the PingAccess agent retrieved from the IIS context. Learn more in Agent inventory logging.	`true`
`agent.inventory`	Specifies additional values to include in the `vnd-pi-agent` agent inventory header. This property uses the following syntax: `agent.inventory=exampleheader=TEST;exampleheader2=TEST2;` **Note** The specified header fields are case-sensitive.	This property isn't present by default.

Property	Definition	Default Value
`agent.cache.defaultTokenType`	Specifies which token-type to favor when making an access decision if both a cookie and an authorization header token are included in a request. Acceptable values are `C` for cookie or `A` for authorization bearer token. Learn more in the **token-type**, **path**, and **vnd-pi-token-cache-oauth-ttl** entries in PAAP agent response in the PingAccess 8.1 documentation. **Note** This property isn't listed in the `agent.properties` file by default. To configure `A` as the `agent.cache.defaultTokenType`, you must add this property to the `agent.properties` file and set it equal to `A`.	`C`
`agent.request.block.xss.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request body. When defining these values, you can: • Use actual characters or URL-encoded characters • Specify a range of characters, such as a-z or `%00-%1f` • Use commas as delimiters to define multiple values **Note** To block a comma, you must URL encode it as `%2C`. • Configure any of the following special combinations for one value: ◦ Two forward slashes (//) ◦ A period and a forward slash (./) ◦ A forward slash and a period (/.) ◦ A forward slash and an asterisk (/*) ◦ An asterisk and a period (*.) The following example demonstrates how to block some common XSS characters: `agent.request.block.xss.characters=<,>,',/\,\,%22,%0a,%0d` **Note** Blocked requests are recorded as error entries in the PingAccess log. To get more details about why a particular request was blocked, set the log level to debug and review these error entries.	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.uri.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request URI. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common URI characters: `agent.request.block.uri.characters=//,./,/.,/,.,~,\,%00-%1f,%7f`	This property isn't present by default.
`agent.request.block.query.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's query parameters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common query characters: `agent.request.block.query.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.form.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's form parameters. **Important** The request must have a `Content-Type` header value of `application/x-www-form-urlencoded` for the agent to block form characters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common form characters: `agent.request.block.form.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.xss.http.status`	Set a custom status code to display when the agent blocks a request because of a bad XSS character. **Tip** When configuring HTTP status codes initially, consider using a `500` error code to create more obvious test results. After you complete testing, set the HTTP status code to a more reasonable value, such as a `400` error code. The following example demonstrates how to set an XSS HTTP status code: `agent.request.block.xss.http.status=400`	This property isn't present by default.
`agent.request.block.uri.http.status`	Set a custom status code to display when the agent blocks a request because of a bad URI character. The following example demonstrates how to set a URI HTTP status code: `agent.request.block.uri.http.status=404`	This property isn't present by default.
`agent.request.block.query.http.status`	Set a custom status code to display when the agent blocks a request because of a bad query character. The following example demonstrates how to set a query HTTP status code: `agent.request.block.query.http.status=400`	This property isn't present by default.
`agent.request.block.form.http.status`	Set a custom status code to display when the agent blocks a request because of a bad form character. The following example demonstrates how to set a form HTTP status code: `agent.request.block.form.http.status=400`	This property isn't present by default.

Tip

You can add comments to the `agent.properties` files if necessary. The agent ignores lines beginning with the `#` or `!` characters.

Important

If you make changes to the `agent.properties` file, you must restart the web server.

> **Tip**
>
> Learn more about improving agent performance in the Performance tuning guide.

Log Configuration

The PingAccess agent for Internet Information Services (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.

The 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 and 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 ahigh volume log and might contain sensitive identity and token information. This log is disabled and hidden by default.

> **Note**
>
> To view and enable a log:
>
> 1. To display all of the logs, go to **View → Show Analytic and Debug Logs** in Event Viewer.
> 2. In the console tree, select the log that you want to enable.
> 3. In the Actions pane, click **Enable Log**.

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 this command.

```
PS> get-winevent -logname PingAccess-Agent/Admin,PingAccess-Agent/Debug,PingAccess-Agent/Analytic -Oldest
```

Rotating a CA

Rotate the certificate authority (CA) used by an agent while minimizing the impact to agent communications.

Steps

1. On the agent web server, update the `agent.properties` file to add the new CA certificate.
    1. Concatenate the old and new CA certificates in PEM encoding format into a new file.

2. Encode the contents of the file to Base64.
3. Open the `agent.properties` file and set the value of the `agent.engine.configuration.bootstrap.truststore` line to the encoded content.

   Example:

   ```
   agent.engine.configuration.bootstrap.truststore=<Encoded_content>
   ```

2. Restart the agent web server.
3. Update the PingAccess configuration to use a new server certificate signed by the new CA for the agent HTTPS listener.
   1. Identify a key pair to use. If necessary, create a new key pair.

      Learn more in Generating new key pairs.

   2. Generate a CSR for that key pair.

      Learn more in Generating certificate signing requests.

   3. Submit that CSR to the new CA to get a new signed certificate.
   4. Import the CSR response (the new certificate) into PingAccess.

      Learn more in Importing certificates.

   5. Assign the key pair to the agent HTTPS listener.

      Learn more in Assigning key pairs to HTTPS listeners.

Troubleshooting

This table lists some potential problems and resolutions you might encounter with the PingAccess agent for Internet Information Services (IIS).

Issue	Resolution
The Installer fails to successfully install the agent.	Use the steps listed in the Manual Installation procedure to validate the installation and to manually complete the installation. Review the MSI installer log file for the installation to identify errors. The log file is stored in the Temp directory `C:\Users[.varname]_<username>_\AppData\Local\Temp` by default. The filename is not fixed, so you must locate the most recent MSI*.log file. 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.

Issue	Resolution
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.

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 Installing on IIS.

Issue	Resolution

Agent receives an unknown protocol error when attempting to contact the administrative node	This can indicate that the operating system is using SHA-1 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 SHA-1:
	1. Open the `run.properties` file.
	2. Add TLSv1 to the protocol list. For example:
	`tls.default.protocols=TLSv1, TLSv1.1, TLSv1.2, TLSv1.3`
	3. Add the SHA entries to the cipher suites list. For example:
	`tls.default.cipherSuites = TLS_CHACHA20_POLY1305_SHA256,\`
	`TLS_AES_256_GCM_SHA384,\`
	`TLS_AES_128_GCM_SHA256,\`
	`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_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`

Validating the IIS Configuration

Verify that an Internet Information Services (IIS) agent has installed successfully.

About this task

For a minimal configuration of the PingAccess agent for IIS, the following steps outline the changes made during installation that might 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.
    1. Run the command `net stop w3svc`.
    2. Run 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`.

    Example:

```
<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`.

   Example:

   ```
   <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 go to **Management → Configuration Editor**.
5. Select the `system.webServer/paa` section and validate that the paths added to `applicationHost.config` are correct.
6. Register the agent module with IIS.
   1. Open IIS Manager, then select the web server the agent is being added to.
   2. Click **Modules**.
   3. Click **Configure Native Modules**.
   4. Click **Register** and enter the following information.

Name	`PingAccessAgentModule`
Path	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\paa-iis-module.dll`

   5. Click **OK**.
   6. Click **OK**.
   7. Run the command `iisreset /restart`.

Manually removing agents on IIS

Manually remove the agent if an attempt to remove the agent from a system fails.

Steps

1. Stop Microsoft IIS.
   1. Run the command `net stop w3svc`.
   2. Run 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`.

   Example:

   ```
   <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`.

   Example:

   ```
   <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 go to **Management → Configuration Editor**.

5. Select the `system.webServer/paa` section and validate that the paths were properly removed from `applicationHost.config`.

6. Deregister the agent module with IIS.

   1. Open IIS Manager, and then select the web server from which the agent is being removed.
   2. Click **Modules**.
   3. Click **Configure Native Modules**.
   4. Select the `PingAccessAgentModule` registered module, and then click **Remove**.
   5. Click **OK**.
   6. Run the command `iisreset /restart`.

PingAccess Agent for IIS release notes

These release notes summarize the changes in current and previous PingAccess agent for Internet Information Services (IIS) updates. Updated July 11, 2024.

PingAccess Agent for IIS 1.5 (July 2024)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.4.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types. For more information, see the **Cache multiple token-types for Web + API applications** entry in the PingAccess 8.1 release notes, and the `agent.cache.defaultTokenType` property on the IIS agent configuration page.

> **Note**
>
> Existing agent environments ignore the new `vnd-pi-token-cache-oauth-ttl` header and additional paths in the `vnd-pi-resource-cache` header.
> To see the performance boost, upgrade to PingAccess 8.1 and upgrade to the latest version of the IIS agent. Otherwise, continue to use an earlier agent version.

Block bad characters in IIS agent deployments

New PAA-251

Configure the PingAccess agent for IIS to block requests that contain bad characters in the URI, query parameters, form parameters, or request body without having to reach out to PingAccess for a decision.

Added eight new properties to the agent:

1. `agent.request.block.xss.characters`
2. `agent.request.block.uri.characters`
3. `agent.request.block.query.characters`
4. `agent.request.block.form.characters`
5. `agent.request.block.xss.http.status`
6. `agent.request.block.uri.http.status`
7. `agent.request.block.query.http.status`
8. `agent.request.block.form.http.status`

Learn more in the IIS agent configuration configuration page.

> **Note**
>
> For large scale or more complex blocking decisions, it's best practice for the agent to reach out to PingAccess for a decision.

Configure the IIS agent to ignore CRL checking if revocation server is unresponsive

Improved PAA-265

Added a new configuration option to give protected applications better reliability without giving up the ability to perform CRL checking when the server is available: the `agent.engine.configuration.checkCertRevocation.bestEffort` property.

This change provides better alignment between PingAccess, PingFederate, and PingAccess policy server CRL checking. Learn more in IIS agent configuration.

PingAccess Agent for IIS 1.4.4 (July 2021)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.3.

Added agent inventory response

New PAA-224

Added agent inventory response.

PingAccess Agent for IIS 1.4.3 (December 2020)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.3.

Disable IIS caching only when modifying response

Improved PAA-202

Updated the agent to only disable IIS caching when the agent modifies the response. This preserves performance while mitigating an IIS session swapping vulnerability.

PingAccess Agent for IIS 1.4.2 (July 2020)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.3.

Fixed application pool crashing

Fixed PAA-194-15776

Fixed an issue that caused intermittent application pool crashes.

PingAccess Agent for IIS 1.4.1 (February 2020)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.2.1.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for IIS 1.4 (June 2019)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.2.0.

Set the policy caching mechanism in `agent.properties`

New PAA-105

Added ability to set policy caching mechanism using a property in the `agent.properties` file.

Manage agent processing for a request based on the note field

New

Added ability to enable or disable agent processing for a request based on a note field.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for IIS 1.3.2 (November 2018)

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess Agent for IIS 1.3 (January 2017)

Agent for SDK compatibility

Improved

Updated to version 1.1.1 of the PingAccess Agent SDK for C.

Use IIS 10 on Windows Server 2016

New

Added support for IIS 10 on Windows Server 2016.

Fixed IIS `Preload Enabled` setting

Fixed

Resolved an issue with the IIS `Preload Enabled` setting.

PingAccess Agent for IIS 1.2.1 (November 2016)

Configure the `Preload Enabled` setting in IIS

New

Added support for the `Preload Enabled` setting in IIS.

Security enhancements

Security

Made agent security enhancements.

PingAccess Agent for IIS 1.2 (August 2016)

Agent SDK for C compatibility

Improved

Updated to version 1.0.1 of the PingAccess Agent SDK for C.

PingAccess Agent for IIS 1.1.2 (February 2016)

Fixed an issue preventing custom request headers from setting

Fixed

Addressed issue with custom request headers not being set when URL contains query string parameters.

PingAccess Agent for IIS 1.1.1 (September 2015)

Use the IIS WebSphere plugin

New

Addressed compatibility with the IIS plugin for WebSphere.

PingAccess Agent for IIS 1.1 (December 2014)

Compatibility with IIS 7.0 running on Windows Server 2008

New

Added Support for Microsoft Internet Information Services (IIS) 7.0 running on Windows Server 2008.

Compatibility with IIS 7.5 running on Windows Server 2008 R2

New

Added Support for Microsoft Internet Information Services (IIS) 7.5 running on Windows Server 2008 R2.

Compatibility with IIS 8.0 running on Windows Server 2012 Datacenter Edition

New

Added Support for Microsoft Internet Information Services (IIS) 8.0 running on Windows Server 2012 Datacenter Edition.

Fixed a potential security issue

Security

Fixed a potential security issue related to caching (SECBL007). This security bulletin is available in the Ping Identity Support Portal (https://support.pingidentity.com/s/).

PingAccess Agent for IIS 1.0 (July 2014)

Initial release

Info

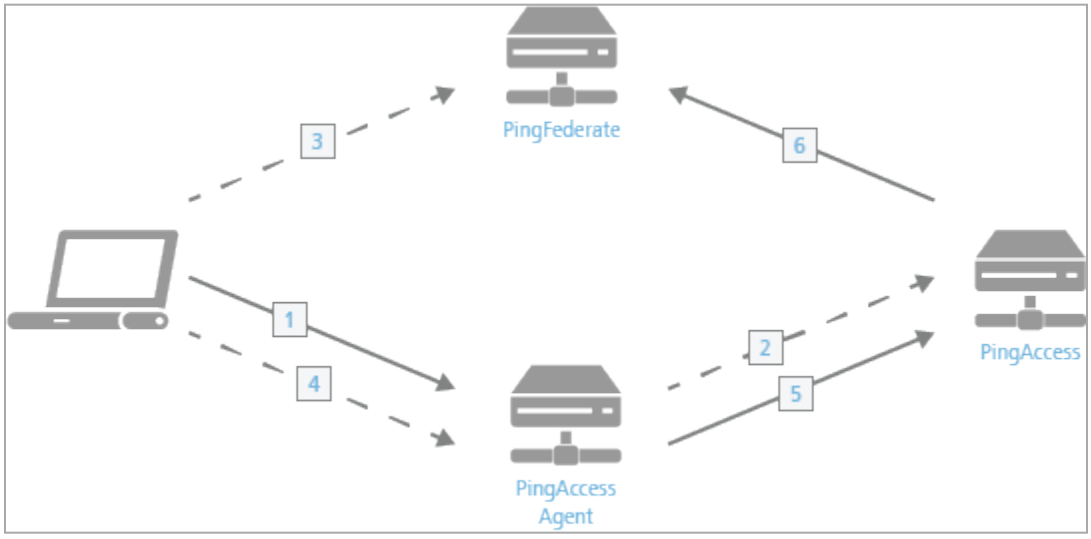
Initial release of the PingAccess Agent for IIS.

PingAccess Agent for NGINX

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.

Note

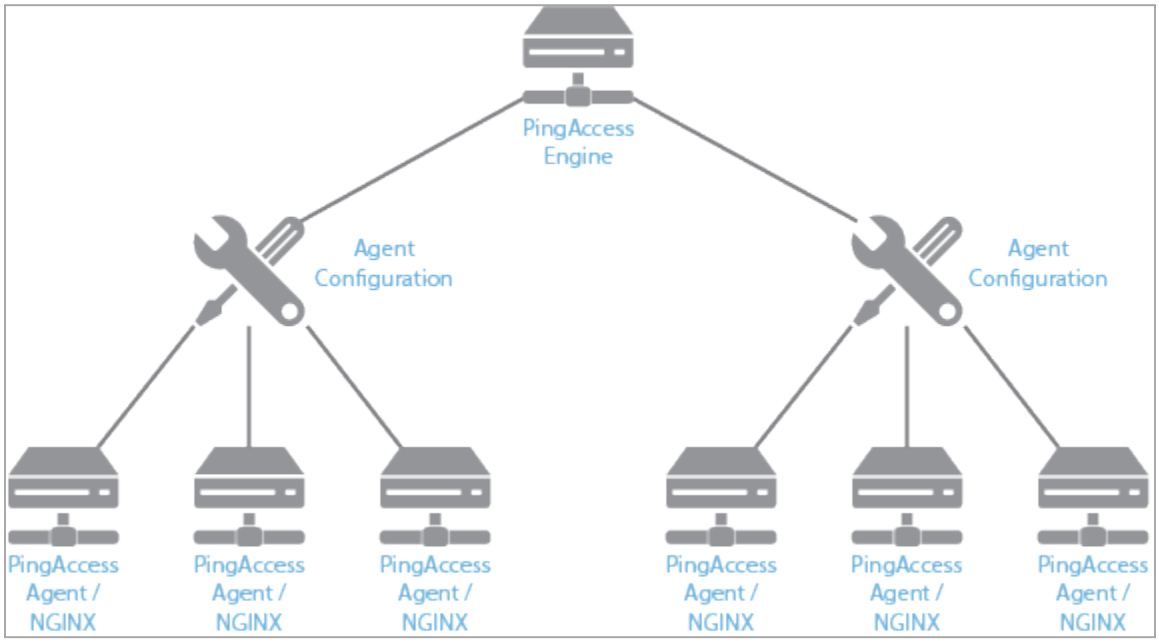
Download the PingAccess agent for NGINX on the PingAccess Downloads page.



When a PingAccess agent is added to the policy decision process:

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 Uniform Resource Locator (URL) policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user signs on, 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 administrative 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.



An agent node is a shared configuration used by one or more agents, rather than a specific agent instance.

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

NGINX agent system requirements

The PingAccess agent for NGINX is supported on these platforms.

Operating System	Compatible NGINX Versions
Red Hat Enterprise Linux (RHEL) Server 7 (x86_64)	• NGINX Plus R31 or NGINX OSS 1.25.3 • NGINX Plus R32 or NGINX OSS 1.25.5
Red Hat Enterprise Linux (RHEL) Server 8 (x86_64)	• NGINX Plus R31 or NGINX OSS 1.25.3 • NGINX Plus R32 or NGINX OSS 1.25.5 • NGINX Plus R33 or NGINX OSS 1.27.2 • NGINX Plus R34 or NGINX OSS 1.27.4
Red Hat Enterprise Linux (RHEL) Server 9 (x86_64)	• NGINX Plus R31 or NGINX OSS 1.25.3 • NGINX Plus R32 or NGINX OSS 1.25.5 • NGINX Plus R33 or NGINX OSS 1.27.2 • NGINX Plus R34 or NGINX OSS 1.27.4

Installing on NGINX

Install a PingAccess agent on an NGINX system.

Before you begin

This procedure makes the following assumptions:

- The PingAccess NGINX agent zip content is extracted to the `$PINGACCESS_AGENT_NGINX` folder.

> **Note**
>
> Amazon Linux 2 systems use the Red Hat Enterprise Linux 7 download bundles.

- 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 might be available through 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 **Applications > Agents**.
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 `$NGINX/paa/agent.properties`.
6. If you are installing on NGINX OSS, edit the `agent.properties` file and set the `agent.engine.configuration.maxConnections` property value to `0`.
7. To load the PingAccess agent for NGINX module, add the following directive to the NGINX configuration file, `$NGINX/nginx.conf`.

```
load_module modules/ngx_http_paa_module.so;
```

8. To configure the PingAccess Agent for NGINX module, add the following directive to the NGINX configuration file, `$NGINX/nginx.conf`, within the http {} block.

```
include $NGINX/paa/http.conf;
```

> **Important**
>
> In PingAccess Manage Agents, **PingAccess Host** must match the certificate CN or Subject Alternative Name (SAN).

9. To enable the PingAccess Agent, modify the following property in the file `$NGINX/paa/http.conf`.

```
paa_enabled on;
```

> **Important**
>
> If `agent.engine.configuration.failover.hosts` in `agent.properties` is set, the `paa_upstream` and upstream blocks in the provided `http.conf` file will need to be updated to a value consistent with a CN or SAN of the certificate associated with the PingAccess Agent HTTPS Listener. Otherwise, you will not be able to establish an HTTPS connection to either the primary or the backup server.

10. If you are installing on NGINX OSS, edit the `$NGINX/paa/http.conf` file and comment out the line containing the queue directive.

    For example:

```
# queue 65536;
```

11. Restart the NGINX server:
    1. To stop the NGINX server, run the following command. `sudo systemctl stop nginx`
    2. To start the NGINX server, run the following command. `sudo systemctl start nginx`

Uninstalling on NGINX

Remove the agent from an NGINX system.

Steps

- Run the following command.

```
sudo yum remove pingaccess-agent-nginx.x86_64
```

NGINX agent configuration

Manage the PingAccess agent for NGINX configuration through the `$NGINX/paa/http.conf` and `agent.properties` configuration files.

The `$NGINX/paa/http.conf` file contains the configuration options defined in the following table.

Parameter	Definition	Default Value
`paa_property_files`	Properties file that stores configuration data used to connect the agent to the PingAccess engine nodes.	`$NGINX/paa/agent.properties`
`paa_enabled`	Determines whether the agent is enabled or disabled for a specific server configuration. Valid values are `on` or `off`. To control which blocks that the agent protects, you can set the `paa_enabled` parameter on: • Specific server blocks within the NGINX server • Specific location blocks For example, if you want to set up an unprotected passthrough resource that PingAccess should always allow access to, you can set `paa_enabled` to `off` in the location block that represents the unprotected resource. This expedites request processing because the agent doesn't need to request a decision from the PingAccess engine. **Note** You can apply this parameter globally to the `http` block. The agent follows the most specific value that you set. If you set the `paa_enabled` parameter to `off` globally, ensure that the `paa_enabled` parameter is set to `on` for the PingAccess reserved application context root. By default, this context root is `/pa`.	`off`

Parameter	Definition	Default Value
`paa_upstream`	Defines the upstream that the PingAccess agent uses to route policy decision requests to PingAccess policy servers.	`pingaccess-policy-server`
`paa_upstream_max_response_header_size`	Defines the maximum size of the response header, in bytes, that the PingAccess agent can receive from a PingAccess policy server.	`4096`
`paa_thread_pool`	Defines the thread pool to use for blocking operations performed by the agent. **Note** This only includes policy cache lookup operations when using the ZeroMQ multiprocess policy cache.	`default`

Note

- You do not have to make any changes to `http.conf` if you followed the PingAccess agent for NGINX Installation steps.
- Changes to the `paa_upstream` parameter will impact how the agent communicates with PingAccess. Incorrect changes might lead to a non-functional agent.
- 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 properties:

Property	Definition	Default Value
`agent.engine.configuration.scheme`	The Uniform Resource Identifier (URI) scheme used to connect to the engine node. Acceptable values are: • `http` • `https`	`https`
`agent.engine.configuration.host`	The PingAccess host name.	The value in the agent node's `PingAccess Host` field.

Property	Definition	Default Value
`agent.engine.configuration.port`	The port that the agent connects to on the PingAccess host. **Tip** This value is defined in the PingAccess run.properties file.	Defined in the PingAccess admin console
`agent.engine.configuration.username`	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess admin console
`agent.engine.configuration.shared.secret`	The password which is used to authenticate the agent to the engine.	Defined in the PingAccess admin console
`agent.engine.configuration.bootstrap.truststore`	The base64-encoded public certificate which is 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, complete the following steps to verify that the Agent Trusted Certificate is configured correctly: 1. Base64 decode the public certificate into a `.crt` file and review the contents. 2. In the PingAccess server, make sure that the agent HTTP listener is using the matching private key. Learn more in Assigning key pairs.	Generated by PingAccess
`agent.engine.configuration.maxConnections`	The number of connections that a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host` property.	`10`
`agent.engine.configuration.timeout`	The maximum amount of time, in milliseconds, that an agent request made to PingAccess can take. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.engine.configuration.connectTimeout`	The maximum amount of time, in milliseconds, that the agent can take to connect to the PingAccess engine. If this time is exceeded, the client receives a generic `500 Server Error` response.	`30000`
`agent.cache.missInitialTimeout`	The maximum amount of time (in milliseconds) that 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 that 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`

Property	Definition	Default Value
`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`
`agent.cache.maxTokens`	The maximum number of tokens that are 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 policy decision caching 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. You might want to use this option 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. Because of the performance penalty, only use this option if necessary.	`0`
`agent.engine.configuration.failover.hosts`	The host name 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 the name `pa.nginx`, set the upstream name to `upstream pa.nginx`.	Defined in the PingAccess admin console
`agent.engine.configuration.failover.failedRetryTimeout`	The number of seconds to wait before the agent should retry connecting to a failed PingAccess server.	`60`
`agent.engine.configuration.failover.MaxRetries`	The number of times to retry a connection to a PingAccess server after an unsuccessful attempt. If all retries fail, the agent marks the PingAccess server as failed for the duration of the `agent.engine.configuration.failover.failedRetryTimeout` value and tries another PingAccess server if one is available.	`2`

Property	Definition	Default Value
`agent.cache.type`	Controls the type of policy cache used by the agent. There are three acceptable values for this property: ***AUTO*** 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*** Does not share policy cache entries across worker processes. ***ZMQ*** Allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	`AUTO`
`agent.send.inventory`	Determines whether the `vnd-pi-agent` agent inventory header is sent along with each request to the PingAccess policy server. This header contains the following fields: **v** The PingAccess agent version. **t** The type of PingAccess agent retrieved using the `NGINX_VER_BUILD` macro. **h** The host name of the PingAccess agent retrieved using the `ServerName` directive. Learn more in Agent inventory logging.	`true`
`agent.inventory`	Specifies additional values to include in the `vnd-pi-agent` agent inventory header. This property uses the following syntax: `agent.inventory=exampleheader=TEST;exampleheader2=TEST2;` **Note** The specified header fields are case-sensitive.	This property isn't present by default.

Property	Definition	Default Value
`agent.cache.defaultTokenType`	Specifies which token-type to favor when making an access decision if both a cookie and an authorization header token are included in a request. Acceptable values are `C` for cookie or `A` for authorization bearer token. Learn more in the **token-type**, **path**, and **vnd-pi-token-cache-oauth-ttl** entries in PAAP agent response in the PingAccess 8.1 documentation. **Note** This property isn't listed in the `agent.properties` file by default. To configure `A` as the `agent.cache.defaultTokenType`, you must add this property to the `agent.properties` file and set it equal to `A`.	`C`
`agent.request.block.xss.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request body. When defining these values, you can: • Use actual characters or URL-encoded characters • Specify a range of characters, such as a-z or `%00-%1f` • Use commas as delimiters to define multiple values **Note** To block a comma, you must URL encode it as `%2C`. • Configure any of the following special combinations for one value: ◦ Two forward slashes (//) ◦ A period and a forward slash (./) ◦ A forward slash and a period (/.) ◦ A forward slash and an asterisk (/*) ◦ An asterisk and a period (*.) The following example demonstrates how to block some common XSS characters: `agent.request.block.xss.characters=<,>,',/\,\,%22,%0a,%0d` **Note** Blocked requests are recorded as error entries in the PingAccess log. To get more details about why a particular request was blocked, set the log level to debug and review these error entries.	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.uri.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request URI. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common URI characters: `agent.request.block.uri.characters=//,./,/.,/,.,~,\,%00-%1f,%7f`	This property isn't present by default.
`agent.request.block.query.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's query parameters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common query characters: `agent.request.block.query.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.
`agent.request.block.form.characters`	If present, specifies a value (or values) that prompts PingAccess to block a request if it finds one or more of them in the request's form parameters. **Important** The request must have a `Content-Type` header value of `application/x-www-form-urlencoded` for the agent to block form characters. When defining these values, follow the syntax established in the `agent.request.block.xss.characters` table entry. The following example demonstrates how to block some common form characters: `agent.request.block.form.characters=<,>,&,%22,%27,%28,%29,%7b,%7d`	This property isn't present by default.

Property	Definition	Default Value
`agent.request.block.xss.http.status`	Set a custom status code to display when the agent blocks a request because of a bad XSS character. **Tip** When configuring HTTP status codes initially, consider using a `500` error code to create more obvious test results. After you complete testing, set the HTTP status code to a more reasonable value, such as a `400` error code. The following example demonstrates how to set an XSS HTTP status code: `agent.request.block.xss.http.status=400`	This property isn't present by default.
`agent.request.block.uri.http.status`	Set a custom status code to display when the agent blocks a request because of a bad URI character. The following example demonstrates how to set a URI HTTP status code: `agent.request.block.uri.http.status=404`	This property isn't present by default.
`agent.request.block.query.http.status`	Set a custom status code to display when the agent blocks a request because of a bad query character. The following example demonstrates how to set a query HTTP status code: `agent.request.block.query.http.status=400`	This property isn't present by default.
`agent.request.block.form.http.status`	Set a custom status code to display when the agent blocks a request because of a bad form character. The following example demonstrates how to set a form HTTP status code: `agent.request.block.form.http.status=400`	This property isn't present by default.

Tip

You can add comments to the `agent.properties` files if necessary. The agent ignores lines beginning with the `#` or `!` characters.

Important

If you make changes to the `agent.properties` file, you must restart the web server.

> **Tip**
>
> Learn more about improving agent performance in the Performance tuning guide.

Rotating a CA

Rotate the CA used by an agent while minimizing the impact to agent communications.

Steps

1. On the agent web server, update the `agent.properties` file to add the new CA certificate.
    1. Concatenate the old and new CA certificates in PEM encoding format into a new file.
    2. Encode the contents of the file to Base64.
    3. Open the `agent.properties` file and set the value of the `agent.engine.configuration.bootstrap.truststore` line to the encoded content.

        Example:

        ```
        agent.engine.configuration.bootstrap.truststore=<Encoded_content>
        ```
2. Restart the agent web server.
3. Update the PingAccess configuration to use a new server certificate signed by the new CA for the agent HTTPS listener.
    1. Identify a key pair to use. If necessary, create a new key pair.

        Learn more in Generating new key pairs.
    2. Generate a CSR for that key pair.

        Learn more in Generating certificate signing requests.
    3. Submit that CSR to the new CA to get a new signed certificate.
    4. Import the CSR response (the new certificate) into PingAccess.

        Learn more in Importing certificates.
    5. Assign the key pair to the agent HTTPS listener.

        Learn more in Assigning key pairs to HTTPS listeners.

PingAccess agent for NGINX release notes

These release notes summarize the changes in current and previous PingAccess agent for NGINX updates.

PingAccess agent for NGINX 3.0 (May 2025)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 3.0.

Authenticate the PingAccess agent for NGINX with a bearer token

New PASDKC-199

Authenticate PingAccess agents to the engine nodes with a stronger authentication method.

> **Note**
>
> To use this feature, you must upgrade to PingAccess 8.2 and the PingAccess agent for NGINX 3.0 or later.

Learn more in the PingAccess 8.2 release notes. You can find setup instructions in Configuring PingAccess agents to use bearer token authentication and SDK for C 3.0.

PingAccess agent for NGINX 2.2 (December 2024)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.4.1.

RHEL 7 deprecation

Info

Support for RHEL 7 will be deprecated in version 1.5.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types. Learn more in the **Cache multiple token-types for Web + API applications** entry in the PingAccess 8.1 release notes, and the `agent.cache.defaultTokenType` property on the NGINX agent configuration page.

> **Note**
>
> Existing agent environments ignore the new `vnd-pi-token-cache-oauth-ttl` header and additional paths in the `vnd-pi-resource-cache` header.
> To see the performance boost, upgrade to PingAccess 8.1 or later and upgrade to the latest version of the NGINX agent. Otherwise, continue to use an earlier agent version.

Block bad characters in NGINX agent deployments

New PAA-251

Configure the PingAccess agent for NGINX to block requests that contain bad characters in the URI, query parameters, form parameters, or request body without reaching out to PingAccess for a decision.

Added eight new properties to the agent:

1. `agent.request.block.xss.characters`
2. `agent.request.block.uri.characters`
3. `agent.request.block.query.characters`
4. `agent.request.block.form.characters`
5. `agent.request.block.xss.http.status`
6. `agent.request.block.uri.http.status`
7. `agent.request.block.query.http.status`
8. `agent.request.block.form.http.status`

Learn more in the NGINX agent configuration page.

> **Note**
>
> For large scale or more complex blocking decisions, it's best practice for the agent to reach out to PingAccess for a decision.

PingAccess agent for NGINX 2.1.1 (November 2020)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.3.

Fixed issues that occurred after a request header modification

Fixed PAA-203

Fixed an issue that caused an invalid memory access and sometimes caused crashes after a request header modification by another NGINX module.

PingAccess agent for NGINX 2.1 (July 2020)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.3.

Added agent inventory callback API

New PAA-177

Added agent inventory callback API.

PingAccess agent for NGINX 2.0.2 (February 2020)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.2.1.

Set the max size of the received response header

New PAA-161

Added a configuration property to set the maximum size of the response header that can be received from a PingAccess policy server.

PingAccess agent for NGINX 2.0.1 (June 2019)

Agent SDK for C compatibility

Info

Compatible with the Agent SDK for C version 1.2.0.

Fixed a potential security issue

Security

Fixed a potential security issue.

PingAccess agent for NGINX 2.0 (February 2019)

Use the NGINX HTTP stack to communicate with PingAccess policy servers

Improved

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

Security

Fixed a potential security issue.

PingAccess agent for NGINX 1.1.1 (July 2017)

Start or stop NGINX using the `systemctl` command

New

Added support for starting and stopping NGINX using the `systemctl` command.

Fixed SSL connectivity issue

Fixed

Resolved issue with SSL connectivity.

PingAccess agent for NGINX 1.1 (March 2017)

NGINX certification requirements

Improved

Updated the agent to meet NGINX certification requirements.

PingAccess agent for NGINX 1.0 (January 2017)

Initial release

Info

Initial release of the PingAccess agent for NGINX.

PingAccess agent protocol

The PingAccess agent protocol (PAAP) is an HTTP-based protocol for communication and interaction between PingAccess and PingAccess agents.

One goal of basing the PAAP on HTTP is to enable an agent, which runs in an HTTP environment, to use concepts and code libraries that are already at its disposal.

> **Note**
>
> The PAAP uses custom status codes and headers for some functions. To avoid potential conflicts, all custom status codes were designed after consulting the Hypertext Transfer Protocol (HTTP) Status Code Registry.
> PAAP headers use the following prefix:
>
> ```
> vnd-pi-
> ```
>
> In this context, `vnd` represents a vendor extension, and `pi` represents Ping Identity.

An agent typically sits in front of a web application or another protected resource on the web server or load balancer, such as Apache or Microsoft IIS. Agents use the PAAP to communicate with a PingAccess server, version 3.0 or later.

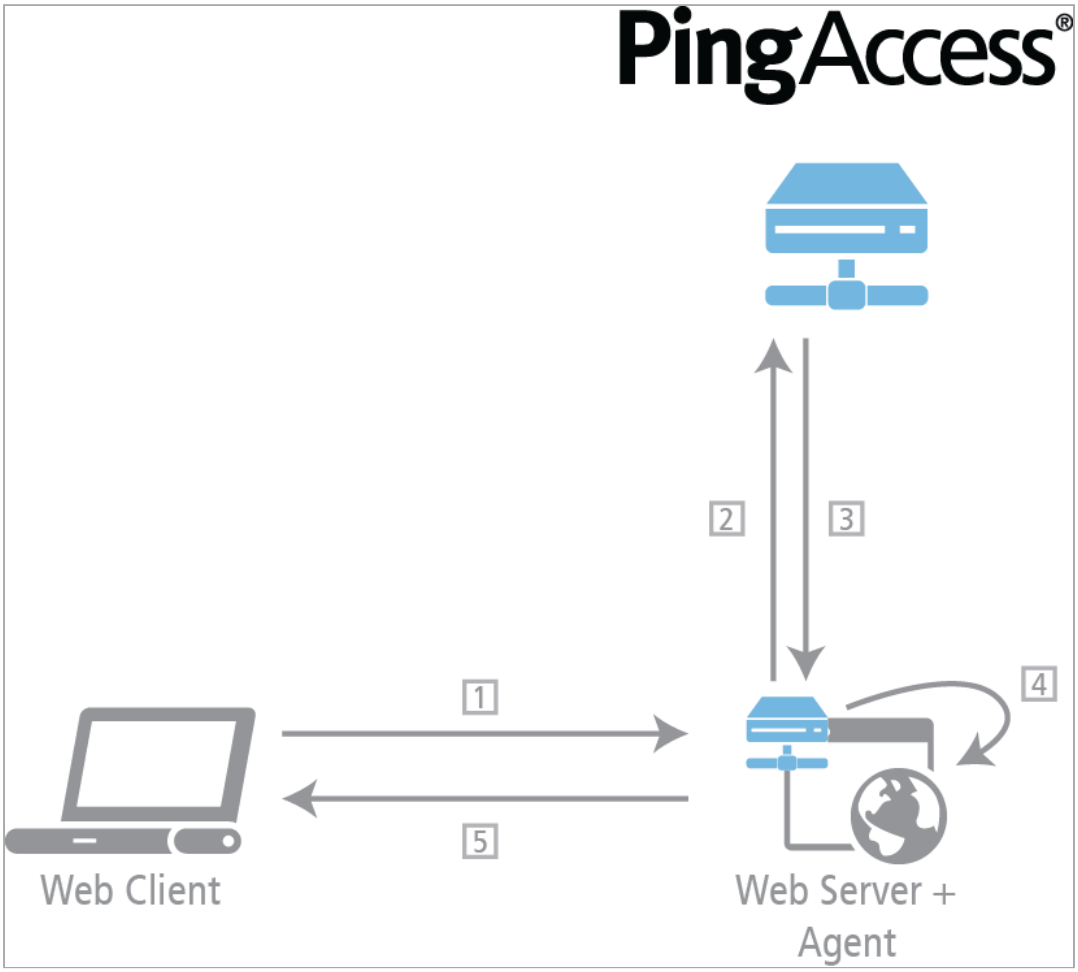
Most responsibilities reside with PingAccess in this model. The intent of the PAAP is to shield agents from configuration and processing details and to maintain policies centrally in PingAccess. This means that agents don't need to know about the signing and encryption keys used by PingAccess or PingFederate.

The PAAP protocol enables you to version and upgrade agents and PingAccess independently of one another.

PingAccess agent protocol flow

The PingAccess agent protocol has a set flow by which requests from clients are evaluated and managed.

The PingAccess agent protocol starts with the client, such as 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 PingAccess. The response from PingAccess instructs the agent whether to allow the original request, as well as any additional actions to take prior to handing it off to the application. It also includes instructions for actions to perform before sending the corresponding response.



Processing steps

1. The client makes a request to the agent system.
2. The agent requests guidance from PingAccess.
3. PingAccess sends a response to the agent.
4. The client request is modified according to any instructions sent by PingAccess.
5. Finally, the client receives a response to their initial request.

PAAP client request

The PingAccess agent protocol (PAAP) 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.

Example

Unauthenticated user request

The following example demonstrates a request 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
```

Example

OIDC Connect flow request

The following example demonstrates a request that's part of the OpenID Connect (OIDC) authentication flow. For brevity, the POST body isn't 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
```

Example

Authenticated resource request

The following example demonstrates 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.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb3JrIiwic3ViIjoiam9lIiwicGhvbmVfbnVtY
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw
```

PAAP agent request

When the agent intercepts a client request, it makes a correlated request to PingAccess to determine if the request is authorized and how to respond.

The agent makes an agent request after it receives a client request to determine what action to take in response. PingAccess then returns an agent response to the agent that communicates how to proceed.

The Request-Line of the agent request is identical to the Request-Line of the client request. Unless otherwise specified, all headers in the agent request are sent as they appear in the client request. The agent request effectively mirrors the client request, except that the message-body of the client request is omitted from the initial agent request.

If PingAccess needs the message-body, PingAccess returns an HTTP `477` response, as defined in PAAP agent response. The message-body is omitted from the initial agent request because PingAccess doesn't need it to make a policy decision in most cases, and removing the message-body makes performance more efficient. The HTTP `477` response mechanism provides PingAccess a way to get the message-body if it's necessary.

HTTP request headers

The following HTTP request headers might require additional agent processing:

Content-Length

The Content-Length header in the agent request has 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 is 0. A subsequent agent request resulting from a `477` response sends 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-agent

This header lets the agent communicate details of its configuration for optional logging purposes. The agent might specify the custom keys specific to the deployment of the agent, or utilize one or more of the well-known keys.

- `v` - the version of the agent making the request.
- `t` - the type of agent and/or the type of platform where the agent resides.
- `h` - the hostname of the server where the agent resides.

These header examples are considered semantically equivalent.

```
vnd-pi-agent: v="1.0.0", h="apache.example.com", t="Apache 2.4.41"
```

```
vnd-pi-agent: v="1.0.0", h="apache.example.com"
vnd-pi-agent: t="Apache 2.4.41"
```

```
vnd-pi-agent: v="1.0.0"
vnd-pi-agent: h="apache.example.com"
vnd-pi-agent: t="Apache 2.4.41"
```

The syntax for the `vnd-pi-agent` value conforms to a dictionary in this specification, https://datatracker.ietf.org/doc/rfc8941/, where member-values are constrained to be an `sh-string` item.

For more information, see Agent inventory logging.

vnd-pi-expect

This header allows the agent to communicate its needs to PingAccess. `!477` is the only defined value, which tells PingAccess that the agent request contains all the data the agent it is capable of sending regarding the client request. PingAccess should never respond with a 477 to an agent request that has `!477` as the value of the `vnd-pi-expect` request header. 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, or 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.

PingAccess 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 PingAccess. 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 PingAccess and the PAA.

In some cases, unrestricted access to the agent protocol at PingAccess might create an information leakage vulnerability. The custom headers returned in the agent response, for example, might reveal internal details of applications or infrastructure that needs to be protected. Potentially worse, the values might reveal content from encrypted Web Access Management (WAM) tokens or reference access tokens. Authenticating PAAs to PingAccess is one means of mitigating the concern.

Authentication is optional. When it is required is at the discretion of PingAccess. Authentication of the agent to PingAccess is intended as a static deployment option, and 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, PingAccess 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 might 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.

```
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.

PingAccess 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.

Example

Policy decision request

The following example demonstrates 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
```

Example

Agent request without POST body

The following example demonstrates an agent request, in this case, the OpenID Connect (OIDC) callback in a web session POST login type. For brevity, the POST body isn't 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
```

Example

Agent request with POST body

The following example demonstrates a policy decision request with the POST body included.

> **Note**
>
> The `vnd-pi-expect: !477` header disallows the HTTP `477` agent response to request the message-body because 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
```

Example

Authenticated user request

The following example request demonstrates an authenticated user trying to access 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.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb3JrIiwic3ViIjoiam9lIiwicGhvbmVfbnVtY
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

After PingAccess receives an agent request, it returns an agent response, which includes an authorization decision and instructions for any additional actions that the agent should perform on the client request or a request for additional information from the agent.

If the agent response contains an HTTP status code that isn't described in this document, PingAccess did not permit the client request. In this case, the content of the agent response, including the status-line, the message-body, and all the headers that aren't named by the header or defined later, is sent back to the client as the content of the client response. This method lets PingAccess:

- Direct the client, when applicable, to PingFederate for user authentication through redirection or even auto-post form.
- 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 to make a policy decision, PingAccess requires the agent to repeat the agent request and include the request body. Following that, the subsequent agent response will include the policy decision and any necessary actions to take.

Currently, PingAccess only requires agents to include the request body in the agent request if it needs to evaluate an auto-posted OpenID Connect Authentication Response. To optimize performance, agents don't include the request body by default. The `477` response provides a means for PingAccess to ask for the request body when necessary.

This enables the agent to leave configuring any callback `redirect_uri` locations to PingAccess. All the agent knows is that it received a `POST` request and that PingAccess asked for the message-body to process it.

After the agent repeats the agent request with the request body included, PingAccess responds to the agent with a generic HTTP response code. PingAccess only returns a `477` response to get an auto-posted OpenID Connect Authentication Response at the redirect Uniform Resource Identifier (URI), so PingAccess never returns a `277` response after a `477` response.

> **Important**
>
> PingAccess should never respond to a `GET` request with a `477`.

HTTP 277 Allowed

A `277` response indicates that 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 `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 include as headers for the client request. If any included headers already exist in the client request, the values are overwritten by the values from the corresponding agent response header. If a named header isn't present in the agent response, it's removed from the client request.

This allows PingAccess to make user attribute information available to the protected application in the headers and to guard against header injection by the client. To guard against malicious header injection by end users, use this mechanism to expose user data to the application. PingAccess should include all header names used in a given context, even if they have no value, so that 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 that are already present in the client request are not overwritten, but new headers are added with the values from the agent response. If a named header isn't 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 to set in the client request as request scoped variables or attributes in an appropriate manner for the environment that the agent resides in. 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. Typically, this is a header that's also named in `vnd-pi-set-req-headers` because those headers generally expose user information to the backend application through headers in the modified client request. `vnd-pi-sub` might name a header that isn't in the `vnd-pi-set-req-headers` list if the agent or the environment in which it's deployed needs to know the username or 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 of those values. If `vnd-pi-sub` names a header that isn't present in the agent response, the agent should ignore it.

vnd-pi-set-resp-headers

The value of this header is a comma-delimited list of header names from the agent response to 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 named header isn't present in the agent response, it's 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 to add to the headers of the client response. Any existing named headers that are already present in the client response are't overwritten, but new headers are added with the values from the agent response. If a named header isn't present in the agent response, no action is taken for that header name in the client response.

This allows PingAccess to set or reset the PingAccess Web Access Management (WAM) token as a cookie in the user's browser.

Generally, for any particular request or response header name, PingAccess should only indicate either a set or an append directive. However, the set directive takes precedence with an agent response. 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. If present, this list implicitly includes the `vnd-pi-omit-resp-headers` header.

Caching

Caching directives aim to improve performance by reducing the number of calls made from the agent to PingAccess.

Resource Definition Caching

When the `vnd-pi-resource-cache` request header is present in the agent request, PingAccess includes the following headers in the agent response. This enables the agent to make initial decisions on how to handle client requests without having to consult PingAccess directly.

vnd-pi-resource-cache

This is a multi-valued header and, in keeping with Section 4.2 of RFC 2616, the values might be comma-delimited, or multiple message-header fields with the same name might be included. Value ordering has a significant impact. When servicing future requests based on the cache, agents evaluate values in the order that 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 paths against which requests are matched. Specify a value of one or more space-delimited quoted strings. Each quoted string is a path value, which can contain wildcards using the asterisk (`*`) character.

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 there isn't a wildcard, the values must match exactly.

> **Note**
>
> If you're using a **Web + API** application, PingAccess adds an additional path to the `vnd-pi-resource-cache` header to cache both cookie and authorization header token-types. For example:
>
> ```
> vnd-pi-resource-cache: path = " /*"; cs = N, kind = P; token-type = C; token-name = <cookie_name>, path = " /*"; cs = N, kind = P; token-type = A; token-name = <OAuth Bearer value>
> ```
>
> Recording both bearer and cookie cache entries for the same path reduces the number of calls that an agent needs to make to PingAccess for access decisions, and enables setting more precise token time-to-live (TTL) values for both token-types.
> When the application type is **Web + API**, PingAccess will only return the TTL header corresponding with the token-type that it used to make the access decision, `vnd-pi-token-cache-ttl` or `vnd-pi-token-cache-oauth-ttl`, even if the request included both a bearer token and cookie.

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 such as `GET`, `POST`, `PUT`, etc. If this component is omitted, all methods are allowed and should match for the resource.

kind

Indicates the kind of resource and the general level of access control protection to be applied to it, such as 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 PingAccess, means that the agent must always make an agent request to PingAccess 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 might 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.

If the client request contains more than one token matching the name and type, the value from the first occurrence must be used as the key to lookup or establish a cache for a particular token and other occurrences must be ignored.

The resource-cache list is valid for and scoped to the host in the client request. When building the resource-cache, PingAccess 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 can 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 PingAccess for new `vnd-pi-resource-cache` and `vnd-pi-resource-cache-ttl` values when the time-to-live on its current resource cache has elapsed.

PingAccess provides configurability over the resource cache `time-to-live` value to balance performance and security goals.

Example

An example `vnd-pi-resource-cache` response header is shown in the following code. 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 PingAccess to determine what to do. Next, it tells PingAccess that requests with a `.jpg`, `.gif`, or `.png` suffix are allowed to pass through. Requests for a path that starts with `/canada/` require a PingAccess WAM token, which will be a cookie named PA.cad. Requests for a path that starts with `/usa/` require a PingAccess WAM token, which will be a cookie named PA.usd. All other requests, indicated by the slash wildcard path, are allowed. 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**
>
> The previous 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 PingAccess 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 PingAccess 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 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 or token-name will likely contain a `vnd-pi-set-req-headers` directive that names non-existent headers 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.

> **Note**
>
> 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 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 a `vnd-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. 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 PingAccess 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 following cache header defined is valid on any agent response, and agents should be prepared to cache all agent responses, rather than just `277` responses.

Generally, 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. For example, 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, so PingAccess enables tuning and configuration options for the TTL directive.

In the event that the token is empty, null, or missing, such as 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`.

> **Note**
>
> If you're using a **Web + API** application, PingAccess returns this TTL header only if it uses the cookie token-type to make an access decision. Otherwise, it will return the `vnd-pi-token-cache-oauth-ttl` header.

vnd-pi-token-cache-oauth-ttl

The value of the resource cache OAuth `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 can be cached and used.

If you're using a **Web + API** application, PingAccess returns this TTL header only if it uses the authorization header token-type to make an access decision. Otherwise, it will return the `vnd-pi-token-cache-ttl` header.

Early Cache Invalidation

PingAccess might include the following header in an agent response to instruct the agent to invalidate its cache. For example, the agent might need to do this 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 and 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 might ignore an invalidation directive for a timestamp that it has already processed.

It is difficult for PingAccess to know the cache state of any particular agent or group of agents. PingAccess can't identify the exact responses that should include the cache invalidation directive. Using the timestamp as the header value allows PingAccess to send the `vnd-pi-cache-invalidated` more indiscriminately while allowing the agent to determine if it has taken, or needs to take, 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 PingAccess might take some time to propagate and might yield a mixed set of old and new behavior.

The invalidation directive set using 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 PingAccess, there are still many requests that necessitate the call and allow the `vnd-pi-cache-invalidated` header to be sent to an agent.

Example

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=aHR0cDovL3JoZWw2NS9hcHBsaWNhdGlvbi9oZWFkZXJzIEFwcGxpY2F0aW9uIFJvb3QrUmVzb3VyY2U&nonce=X18Sm8qrBQ8n0K-0sWnlT1
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
```

Example

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
```

Example

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.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb3JrIiwic3ViIjoiam9lIiwicGhvbmVfbnVtY
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
```

Example

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 forwarded to the application, or allowed to continue in the HTTP processing pipeline of the environment in which the agent is deployed.

Example

Additional HTTP headers

The following example shows the additional HTTP headers added as specified by PingAccess.

```
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.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb3JrIiwic3ViIjoiam9lIiwicGhvbmVfbnVtY
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.

Example

Pass-through from agent response

The following client response shows the direct pass-through of the status code and headers from the agent response:

```
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=aHR0cDovL3JoZWw2NS9hcHBsaWNhdGlvbi9oZWFkZXJzIEFwcGxpY2F0aW9uIFJvb3QrUmVzb3VyY2U&nonce=X18Sm8qrBQ8n0K-0sWnlT1
gs0lMY&scope=openid%20profile%20address%20email%20phone
Set-Cookie: nonce=b000c6a2-4a03-4bde-be29-956456cd1d2a; Path=/; HttpOnly
```

For a related example, see the agent response OpenID Connect (OIDC) authentication example.

Example

Client response with redirect

The following client response example shows the direct pass-through of the status code and headers from the agent response:

```
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.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb3JrIiwic3ViIjoiam9lIiwicGhvbmVfbnVtY
PingAccess 3.x SpecificationAgent Protocol Specification V1.0Page 25
JmZW1hbGUiLCJwcm9maWxlIjoiaHR0cHM6XC9cL3d3dy5waW5naWRlbnRpdHkuY29tXC9wcm9kdWN0c1wvcGluZ2ZlZGVyYXRlXC8iLCJ3ZWJzaXRlIjo
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw; Path=/; HttpOnly
Set-Cookie: nonce=; Path=/; Expires=Thu, 01-Jan-1970 00:00:00 GMT
```

For a related example, see the agent response redirect example.

PingAccess Agent SDK for C

This documentation provides technical guidance for using the PingAccess Agent SDK for C. 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 Software Development Kit (SDK) within a sample agent for Apache.

Additional documentation

The SDK documentation provides detailed reference information for developers. After extracting the `pingaccess-agent-c-sdk-<version>.zip` package, access the application programming interface (API) documentation with a web browser by viewing the file `AGENT_SDK_C_HOME/apidocs/index.html`. Alternatively, find the current version of the API documentation online at https://www.pingidentity.com/content/dam/developer/documentation/pingaccess/agent-c-sdk/1-1-4/apidocs/index.html

Introduction

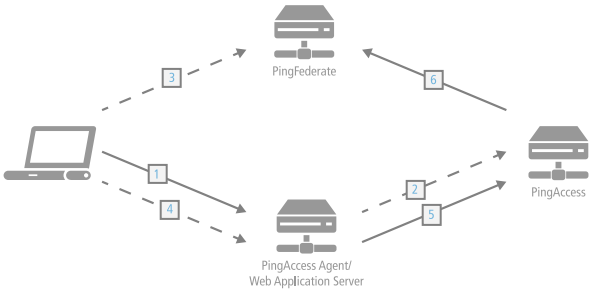
The PingAccess Agent SDK for C provides an API and sample code to enable developers to build agents for C or C++-based application and web servers.

Supported platforms include:

- Red Hat Enterprise Linux Server 7 (32 bit)
- Red Hat Enterprise Linux Server 8 (32 bit or 64 bit)
- SUSE Linux Enterprise Server 12 SP2 (64 bit)
- Microsoft Windows Server 2012
- Microsoft Windows Server 2016

- Microsoft Windows Server 2019

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.



Processing steps

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 redirects the client to PingFederate to establish a session.
3. The user signs on, and PingFederate creates the session.
4. The client is 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 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 Software Development Kit (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 7/8 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 7 (RHEL7), 64-bit
> - Red Hat Enterprise Linux 8 (RHEL8), 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. Further hardening of the Apache server configuration or of the sample configuration file might be required.

If you need assistance using the PingAccess Agent SDK for C, visit the Ping Identity Support Center for help you with your application. Engage the Ping Identity Professional Services team for assistance with developing customizations.

To download the SDK, go to the PingAccess downloads site and click the **Add-ons** tab.

Agent SDK for C directory structure

The PingAccess Agent SDK for C directory contains these subdirectories.

`/`

This directory contains the Agent Software Development Kit (SDK) for C `README.md`, which contains information developers need 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 7 and 8, 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`.

PingAccess Agent SDK for C release notes

These release notes summarize the changes in current and previous PingAccess Agent SDK for C updates.

> **Note**
>
> The PingAccess Agent SDK for C no longer supports FreeBSD 8.

Agent SDK for C 3.0 (April 2025)

RHEL 7 and SLES 12 deprecation

Info

As of Agent SDK for C 3.0, support for RHEL 7 and SLES 12 has been removed.

Create signed JWTs for agent authentication

New PASDKC-197

The agent SDK for C now supports authenticating PingAccess agents to the engine nodes with a bearer token.

> **Important**
>
> To use bearer token authentication, you must upgrade to PingAccess 8.2 or later and either the PingAccess agent for Apache (RHEL), the PingAccess agent for Apache (SLES), or the PingAccess agent for NGINX 3.0.
> Compatibility for the other agent types will be added in a future release.

After you configure a compatible PingAccess agent with the updated `agent.properties` file and select **Require Token Authentication** in the agent's configuration, the agent creates, signs, and sends a unique JWT for every authentication request.

> **Important**
>
> The JWT expires after 2 minutes, so you must ensure you synchronize the agent and the PingAccess server's clocks.

Learn more in the PingAccess 8.2 release notes. You can find setup instructions in Configuring PingAccess agents to use bearer token authentication.

Agent SDK for C 1.4.1 (December 2024)

RHEL 7 deprecation

Info

Support for RHEL 7 will be deprecated in the next version.

Fixed an issue with sending requests to the PingAccess engine

Fixed PASDKC-195

Fixed an issue that caused agents to fail to contact the PingAccess engine about requests meant for the PingAccess reserved application if the root resource was anonymous.

Fixed an issue with form character blocking

Fixed PASDKC-193

Fixed an issue that caused errant form character blocking if XSS blocking was configured. This issue was applicable even if form blocking wasn't configured.

Agent SDK for C 1.4 (October 2024)

Added support for RHEL 9

New

Added support for RHEL 9.

Cache multiple token-types for Web + API applications

New PA-15516

If you use a **Web + API** application, the `vnd-pi-resource-cache` PingAccess agent protocol (PAAP) header now contains an additional path so **Web + API** applications can cache both cookie and authorization header token-types.

Learn more in the **Cache multiple token-types for Web + API applications** entry in the PingAccess 8.1 release notes.

Block bad characters

New PAA-251

Configure an agent to block requests that contain bad characters in the URI, query parameters, form parameters, or request body without having to reach out to PingAccess for a decision.

Added eight new properties to each agent:

1. `agent.request.block.xss.characters`
2. `agent.request.block.uri.characters`

3. `agent.request.block.query.characters`
4. `agent.request.block.form.characters`
5. `agent.request.block.xss.http.status`
6. `agent.request.block.uri.http.status`
7. `agent.request.block.query.http.status`
8. `agent.request.block.form.http.status`

> **Note**
>
> For large scale or more complex blocking decisions, it's best practice for the agent to reach out to PingAccess for a decision.

Ignore CRL checking if revocation server is unresponsive

Improved PAA-265

Added a new configuration option to give protected applications better reliability without giving up the ability to perform CRL checking when the server is available: the `agent.engine.configuration.checkCertRevocation.bestEffort` property.

This change provides better alignment between PingAccess, PingFederate, and PingAccess policy server CRL checking.

> **Important**
>
> To use the `agent.engine.configuration.checkCertRevocation.bestEffort` property, you must be using the native Windows SSL library, Secure Channel (Schannel).

Agent SDK for C 1.3 (June 2020)

Removed support for RHEL 6

Info

Removed support for RHEL 6.

Added support for RHEL 8

New

Added support for RHEL 8.

Added agent inventory callback API

New

Added agent inventory callback API.

Agent SDK for C 1.2.1 (February 2020)

Fixed a potential security issue

Security

Fixed a potential security issue.

Agent SDK for C 1.2 (June 2019)

Fixed a potential security issue

Security

Fixed a potential security issue.

Agent SDK for C 1.1.5 (February 2019)

Added support for FreeBSD 8

New

Added support for FreeBSD 8.

Agent SDK for C 1.1.4 (October 2018)

Fixed potential security issues

Security

Fixed potential security issues.

Agent SDK for C 1.1.3 (August 2018)

Updated libcurl version

Improved

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

Security

Fixed a potential security issue.

Agent SDK for C 1.1.2 (March 2017)

Expanded SUSE Linux Enterprise support

Improved

Added support for:

- SUSE Linux Enterprise Server 11 SP4 (x86_64)
- SUSE Linux Enterprise Server 12 SP2 (x86_64)

Agent SDK for C 1.1.1 (January 2017)

Workaround for Network Security Services library known issue

Info

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 issue with duplicate headers leading to blocked requests

Fixed

Fixed an issue where duplicate headers were included in the backend request to the PingAccess engine, causing the agent to block the request for content.

Agent SDK for C 1.1 (November 2016)

Added policy server failover support

New

Added policy server failover support. Policy server failover support is only provided by the SDK when using the libcurl HTTP client.

Agent SDK for C 1.0.2 (September 2016)

Fixed missing `CRL Distribution Point` extension

Fixed

Fixed an issue where agents could not communicate with PingAccess servers using a certificate signed by a certificate authority because the `CRL Distribution Point` extension is missing. This issue is limited to agents on Windows deployments.

Addressed potential security vulnerability affecting Windows deployments

Security

Addressed a potential security vulnerability. This issue is limited to Windows deployments.

Agent SDK for C 1.0.1 (May 2016)

Fixed ZeroMQ policy cache issue with terminated processes

Fixed

Fixed an issue with ZeroMQ policy cache where a terminated process could cause a condition that resulted in unexpected CPU utilization.

Agent SDK for C 1.0 (April 2016)

Initial release

Info

Initial release of the Agent SDK for C.

PingAccess Agent SDK for Java

This document provides technical guidance for using the PingAccess Agent SDK for Java. 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

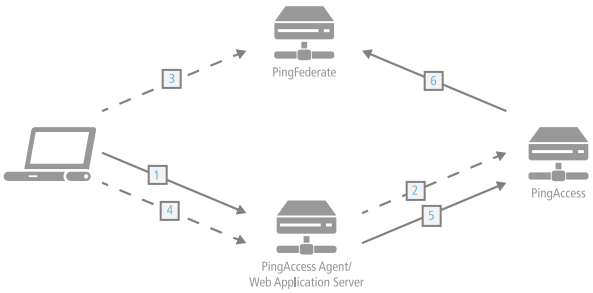
Application developers and system administrators responsible for implementing a Java PingAccess agent should be familiar with Java software-development principles and practices. It describes the use of the Software Development Kit (SDK) within a sample Java Servlet Filter.

Additional documentation

The Java Agent API Javadocs provides detailed reference information for developers. After extracting the `pingaccess-agent-java-sdk-1.1.3.zip` package, the Javadocs is 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 applications and web servers. Agents provide access management features to their containing server by relying on central PingAccess servers over the PingAccess Agent Protocol.



Processing steps:

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 signs on, and PingFederate creates the session.
4. The client is 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 might be required.

If you need assistance using the PingAccess Agent SDK for Java, visit the Ping Identity Support Center to see how we can help you with your application. You can engage the Ping Identity Global Client Services team for assistance with developing customizations.

Downloading the SDK

To download the SDK, go to the PingAccess downloads site and click the **Add-ons** tab.

Agent SDK directory structure

The PingAccess Agent SDK for Java directory contains the following directories.

`/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

Verify that your system meets these prerequisites before installing the PingAccess Agent SDK for Java.

Before you start, ensure you have the Java SDK, Apache Maven and an application server, such as 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, after 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 extracted `pingaccess-agent-java-sdk-1.1.3.0` directory.

To download the SDK, go to the PingAccess downloads site and click the **Add-ons** tab.

Installing the servlet filter sample

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`. See the comments in `agent-sample/WEB-INF/web.xml` for more detail.

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 (OIDC) workflow.
3. In the Tomcat `bin` directory, create a script called `setenv.sh` (Linux) or `setenv.bat` (Windows) with the following contents:

   Choose from:

   - 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 might 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_DHE_RSA_WITH_AES_128_GCM_SHA256

8. Start Tomcat.
9. Open a browser and go 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 notes

These release notes summarize the changes in current and previous PingAccess Agent SDK for Java updates. Updated April 15, 2024.

Agent SDK for Java 1.1.5 (April 2024)

Updated third-party dependencies

Improved PASDKJ-21

Updated third-party dependencies.

Agent SDK for Java 1.1.4 (December 2021)

Upgraded Log4j to version 2.16

Improved

Upgraded Log4j used in the SDK sample implementation to version 2.16.

Agent SDK for Java 1.1.3 (July 2018)

Fixed site cookies setting improperly

Fixed

Fixed an issue where site cookies were not being set properly for HTTP redirects in the servlet container environment (Tomcat).

Fixed error caused by calling a header that doesn't exist

Fixed

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.

Agent SDK for Java 1.1.2 (June 2017)

Fixed incorrect handling of the `set-cookie` header

Fixed

Fixed an issue where the Java agent was handling the PingAccess `set-cookie` header incorrectly.

Fixed issue processing of multiple `set-cookie` headers

Fixed

Fixed an issue where the Java agent wasn't correctly processing multiple `set-cookie` headers sent by PingAccess.

Fixed discrepancy between cookie request headers returned from `getHeader*` versus `getCookies`

Fixed

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.

Agent SDK for Java 1.1.1 (April 2017)

Fixed an issue with unknown attributes

Fixed

Fixed an issue where unknown attributes should be ignored.

Fixed handling of percent-encoded sequences

Fixed

Fixed an issue where percent-encoded sequences in resource paths were being handled incorrectly.

Fixed `Index out of bounds` exception

Fixed

Fixed an issue where an `Index out of bounds` exception occurs if `""` is a cookie value.

Agent SDK for Java 1.1 (August 2015)

Fixed OAuth API response header trimming

Fixed

Fixed an issue where OAuth API response headers were getting trimmed.

Fixed enforced requirement of a username and shared secret

Fixed

Fixed an issue where the Java Agent enforced the requirement of a username and shared secret.

Fixed incorrect `477` response handling

Fixed

Fixed an issue where the Agent was not handling a `477` response correctly.

Agent SDK for Java 1.0 (June 2015)

Initial release

Info

Initial release of the Agent SDK for Java.

PingAccess Add-on SDK for Java

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, integrates 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 to see how we can help you with your application. You can 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.

For more information, see the following topics:

- SDK directory structure
- SDK prerequisites
- Installing the SDK samples

SDK directory structure

The PingAccess SDK directories, `<PA_HOME>/sdk` and `<PA_HOME>/deploy`, contain these files and directories.

Deploy directory

The `<PA_HOME>/deploy` directory is created as a location for all third-party JAR files. PingAccess does not automatically generate any contents for this directory, but any files you place in it are automatically migrated during an upgrade.

The contents of the `<PA_HOME>/deploy` directory are loaded by the `run.sh` or `run.bat` command.

SDK directory

The `<PA_HOME>/sdk` directory contains these files and directories.

File/Directory	Description
`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 can 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. Use these examples for developing your own implementations.
`/samples/SiteAuthenticator/README.md`	Contains the details of the `SiteAuthenticator` samples.
`/samples/IdentityMappings`	Contains a maven project with example plug-in implementations for identity mappings. 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. 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. Use these examples for developing your own implementations.
`/samples/LocaleOverrideService/README.md`	Contains the details of the `LocaleOverrideService` samples.

File/Directory	Description
`/apidocs/`	Contains the SDK Javadocs. To get started, open `index.html`.

SDK prerequisites

The following prerequisites must be met before using the Add-on 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`.

> **Note**
>
> The Ping Identity Maven repository cannot be accessed through a browser because it is designed solely for backend use. To use it, add it to your Maven configuration. For example, including this code in the Maven `pom.xml` file adds the Ping Identity repository to your Maven configuration:
>
> ```
> <repositories>
>     <repository>
>         <releases>
>             <enabled>true</enabled>
>             <updatePolicy>always</updatePolicy>
>             <checksumPolicy>warn</checksumPolicy>
>         </releases>
>         <id>PingIdentityMaven</id>
>         <name>PingIdentity Release</name>
>         <url>http://maven.pingidentity.com/release/</url>
>         <layout>default</layout>
>     </repository>
> </repositories>
> ```

If Internet access is unavailable, update the `pingaccess-sdk` dependency in your `pom.xml` to point to the local installation.

Replace <*PA_HOME*> with the path to the PingAccess 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>
```

Installing the SDK samples

Install rule and site authenticator SDK samples.

Before you begin

Ensure you have the Java SDK and Apache Maven installed.

About this task

Each sample type is installed separately:

- For the rules samples, go to `<PA_HOME>/sdk/samples/Rules`.
- For the site authenticators samples, go to `<PA_HOME>/sdk/samples/SiteAuthenticator`.

Steps

- 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.

Example

```
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

Create your own plugins from scratch using the Add-on 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, to create a rule, 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, to implement 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, to implement a custom rule, create a file called `<PLUGIN_HOME>/META-INF/services/com.pingidentity.pa.sdk.policy.RuleInterceptor`. Its contents are 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>/deploy`.

   > **Important**
   >
   > After copying a custom plugin JAR to the PingAccess lib, you must restart PingAccess 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.

Integrate with third-party services

The Add-on SDK includes the ability for a custom plugin to integrate with external, third-party services using HTTP.

This section provides a high-level overview of utilizing this functionality from a custom plugin:

- Obtaining the HTTP client instance
- Obtaining a handle to a third-party service
- Making a HTTP call to a third-party service

- Base classes
- Sample plugins

Obtaining the HTTP client instance

PingAccess provides access to a HTTP client utility interface, HttpClient, through 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 application programming interface (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 through 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.

```
private static class Configuration extends SimplePluginConfiguration
{
    // ... other code omitted ...

    @UIElement(order = 30,
            type = ConfigurationType.SELECT,
            label = "Risk Authorization Service",
            modelAccessor = ThirdPartyServiceAccessor.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 `ThirdPartyServiceAccessor`.
- 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 OpenID Connect (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 Uniform Resource Identifier (URI) in the OIDC provider metadata. For example, the following 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.

```
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, see 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 using 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.

This information is not exhaustive. Consult the Javadocs to find more details about interfaces discussed here as well as additional functionality.

> **Important**
>
> You must restart PingAccess 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:

1. The rule annotation on the implementation class of the RuleInterceptor is interrogated to determine which PluginConfiguration instance will be instantiated.
2. 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.
3. `PluginConfiguration.setName(String)` is called.
4. PingAccess attempts to map the incoming JavaScript Object Notation (JSON) configuration to the PluginConfiguration instance. The JSON plugin configuration must contain a JSON member for each field, regardless of implied value. Failure to do so can lead to errors.
5. `ConfigurablePlugin.configure(PluginConfiguration)` is called.
6. `Validator.validate(Object, Class[])` method is invoked and provided to the RuleInterceptor.
7. 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, do not use Spring 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`
- `com.pingidentity.pa.sdk.accessor.tps.ThirdPartyServiceModel`
- `com.pingidentity.pa.sdk.http.client.HttpClient`

Differences between rules for agents and sites

Rules can 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 Add-On SDK for Java Migration Guide

When upgrading PingAccess, review the changes made to the PingAccess add-on SDK for Java, analyze your addons, and make any necessary changes to ensure continued compatibility.

The following sections provide a detailed description of the changes, organized by package. Where relevant, code examples show you how to port existing code to account for the changes in the Software Development Kit (SDK) APIs.

> **Important**
>
> Because the SDK for PingAccess 6.1 uses Java 8 features, plugins built against the Java add-on SDK for PingAccess 6.1 or later must be built with JDK 8.

Prevent modification to request in response chain

Starting in PingAccess 6.1, 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 using the administrative application programming interface (API) by annotating a field in the plugin's PluginConfiguration class with a JsonDeserialize annotation, specifying the appropriate custom deserializer from the SDK.

```
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 PingAccess 5.0

```
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 PingAccess 5.0

```
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 declare the validation to be applied to fields in a PluginConfiguration implementation, ensuring the configuration is 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 now validates 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

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.

com.pingidentity.pa.sdk.http.BodyAs the updated Javadocs 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 the plugin code is updated, evaluate whether the body object needs to be used by the plugin.

Using the Body#read method

Before PingAccess 5.0:

```
private void invokeRead(Body body) throws IOException
{
    body.read();
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void invokeGetContent(Body body) throws IOException
{
    byte[] content = body.getContent();
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void invokeGetBodyAsStream(Body body) throws IOException
{
    InputStream stream = body.getBodyAsStream();
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void invokeWrite(Body body, BodyTransferrer bodyTransferrer) throws IOException
{
    body.write(bodyTransferrer);
}
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
private void invokeGetLength(Body body) throws IOException
{
    int length = body.getLength();
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void invokeGetRaw(Body body) throws IOException
{
    byte[] rawBody = body.getRaw();
}
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
private void invokeContinuousBody(BodyFactory bodyFactory, byte[] content)
{
    Body body = bodyFactory.continuousBody(content);
}
```

After PingAccess 5.0:

```
private void invokeContinuousBody(BodyFactory bodyFactory, byte[] content)
{
    Body body = bodyFactory.createInMemoryBody(content);
}
```

Before PingAccess 5.0:

```
private void invokeContinuousBody(BodyFactory bodyFactory, InputStream in)
{
    Body body = bodyFactory.continuousBody(in);
}
```

After PingAccess 5.0:

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 through properties has been enhanced to make it easier to write type-safe code when working with Exchange properties.

Using the Exchange#getCreationTime method

Before PingAccess 5.0:

```
Calendar creationTime = exchange.getCreationTime();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
List<String> destinations = exchange.getDestinations();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
String originalHostHeader = exchange.getOriginalHostHeader();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
String host = exchange.getOriginalHostHeaderHost();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
String port = exchange.getOriginalHostHeaderPort();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
String originalRequestBaseUri = exchange.getOriginalRequestBaseUri();
```

After PingAccess 5.0:

This functionality is no longer supported. A possible replacement is as follows:

```
String originalRequestBaseUri = exchange.getUserAgentProtocol() +
                                "://" +
                                exchange.getUserAgentHost();
```

Using the Exchange#getProperties method

Before PingAccess 5.0:

```
Map<String, String> properties = exchange.getProperties();
```

After PingAccess 5.0:

This functionality is no longer supported. Properties should be obtained individually from the Exchange.

Using the Exchange#getRequestBaseUri method

Before PingAccess 5.0:

```
String requestBaseUri = exchange.getRequestBaseUri();
```

After PingAccess 5.0:

This functionality is no longer supported. A possible replacement is as follows.

```
String requestBaseUri = exchange.getUserAgentProtocol() +
                        "://" +
                        exchange.getUserAgentHost();
```

Using the Exchange#getRequestScheme method

Before PingAccess 5.0:

```
String requestScheme = exchange.getRequestScheme();
```

After PingAccess 5.0:

This functionality is no longer supported. A possible replacement is as follows.

```
String requestScheme = exchange.getUserAgentProtocol() + "://";
```

Using the Exchange#getUser method

Before PingAccess 5.0:

```
private void invokeSetUser(Exchange exchange, User user)
{
    exchange.setUser(user);
}
```

After PingAccess 5.0:

This functionality is no longer supported. The identity associated with an Exchange cannot be replaced.

Using the Exchange#setUser method

Before PingAccess 5.0:

```
private void invokeSetUser(Exchange exchange, User user)
{
    exchange.setUser(user);
}
```

After PingAccess 5.0:

This functionality is no longer supported. The identity associated with an Exchange cannot be replaced.

Using the Exchange#setSourceIp method

Before PingAccess 5.0:

```
private void invokeSetSourceIp(Exchange exchange, String sourceIp)
{
    exchange.setSourceIp(sourceIp);
}
```

After PingAccess 5.0:

This functionality is no longer supported. This value cannot be changed.

Using the Exchange#setProperty method

Before PingAccess 5.0:

```
private void invokeSetProperty(Exchange exchange, String propertyKey, String value)
{
    exchange.setProperty(propertyKey, value);
}
```

After PingAccess 5.0:

```
private void invokeSetProperty(Exchange exchange,
                               ExchangeProperty<String> propertyKey,
                               String value)
{
    exchange.setProperty(propertyKey, value);
}
```

See the Javadocs for ExchangeProperty for instructions on creating an ExchangeProperty object.

Using the Exchange#getProperty method

Before PingAccess 5.0:

```
private void invokeGetProperty(Exchange exchange, String propertyKey)
{
    Object propertyValueObj = exchange.getProperty(propertyKey);
    if (propertyValueObj instanceof String)
    {
        String propertyValue = (String) propertyValueObj;
    }
}
```

After PingAccess 5.0:

```
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 using a HeadersFactory obtained from the ServiceFactory#headersFactory method. The majority of methods on Header have counterparts on the Headers interface. See the Javadocs 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 PingAccess 5.0:

```
private HeaderField createHeaderField(String line)
{
    return new HeaderField(line);
}
```

After PingAccess 5.0:

```
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 avoid having to parse an HTTP header field line.

Using the HeaderField#setHeaderName method

Before PingAccess 5.0:

```
private void invokeSetHeaderName(HeaderField field)
{
    field.setHeaderName(new HeaderName("X-Custom"));
}
```

After PingAccess 5.0:

This functionality is no longer supported. A HeaderField's name is set upon construction and cannot be changed.

Using the HeaderField#getApproximateSize method

Before PingAccess 5.0:

```
int approximateSize = field.getApproximateSize();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
Date date = headers.getDate();
```

After PingAccess 5.0:

```
Date date = Date.from(headers.getDate());
```

Using the Headers#setDate method

Before PingAccess 5.0:

```
private void invokeSetDate(Headers headers, Date date)
{
    headers.setDate(date);
}
```

After PingAccess 5.0:

```
private void invokeSetDate(Headers headers, Date date)
{
    headers.setDate(date.toInstant());
}
```

Using the Headers#getLastModified method

Before PingAccess 5.0:

```
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 PingAccess 5.0:

```
Date lastModifiedDate = Date.from(headers.getLastModified());
```

Using the Headers#setLastModified method

Before PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void invokeSetLastModified(Headers headers, Date date)
{
    headers.setLastModified(date.toInstant());
}
```

com.pingidentity.pa.sdk.http.HeadersFactory

Using the HeadersFactory#createFromRawHeaderFields method

Before PingAccess 5.0:

```
private void invokeCreateFromRawHeaderFields(HeadersFactory factory,
                                             List<String> fields) throws ParseException
{
    Headers headers = factory.createFromRawHeaderFields(fields);
}
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
String localizationKey = status.getLocalizationKey();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
private void createMediaType(String mediaTypeString)
{
    MediaType mediaType = new MediaType(mediaTypeString);
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
InputStream bodyStream = message.getBodyAsStream();
```

After PingAccess 5.0:

This functionality is no longer supported. However, the following code snippet can 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, enable a plugin to propagate errors as an AccessException instead of as a RuntimeException.

Using the Message#getCharset method

Before PingAccess 5.0:

```
Charset charset = message.getCharset();
```

After PingAccess 5.0:

This functionality is no longer supported. However, the following code snippet can 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 might lead to issues in some cases.

Using the Message#getHeader method

Before PingAccess 5.0:

```
Header header = message.getHeader();
```

After PingAccess 5.0:

This functionality is no longer supported. Instead, use Message#getHeaders and the Headers interface instead of Header.

Using the Message#setHeader method

Before PingAccess 5.0:

```
private void invokeSetHeader(Message message, Header header)
{
    message.setHeader(header);
}
```

After PingAccess 5.0:

This functionality is no longer supported. Instead, use Message#setHeaders and the Headers interface instead of Header.

Using the Message#isDeflate method

Before PingAccess 5.0:

```
boolean deflate = message.isDeflate();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
boolean gzip = message.isGzip();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
boolean http10 = message.isHTTP10();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
boolean http11 = message.isHTTP11();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
private void invokeRead(Message message,
                        InputStream inputStream,
                        boolean createBody) throws IOException
{
    message.read(inputStream, createBody);
}
```

After PingAccess 5.0: 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, Uniform Resource Identifier (URI), headers and body. A response attached to an exchange can be replaced by using Exchange#setResponse.

Using the Message#setVersion method

Before PingAccess 5.0:

```
private void invokeSetVersion(Message message, String version)
{
    message.setVersion(version);
}
```

After PingAccess 5.0:

This functionality is no longer supported. The version of a message cannot be changed.

Using the Message#write method

Before PingAccess 5.0:

```
private void invokeWrite(Message message,
                         OutputStream output) throws IOException
{
    message.write(output);
}
```

After PingAccess 5.0:

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_1));
    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 PingAccess 5.0:

```
Method get = Methods.GET
```

After PingAccess 5.0:

```
Method get = Method.GET;
```

Using the Method#getMethodName method

Before PingAccess 5.0:

```
String methodName = method.getMethodName();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void invokeGetPostParams(Request request) throws IOException
{
    Map<String, String[]> postParams = request.getPostParams();
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
boolean multipartFormPost = request.isMultipartFormPost();
```

After PingAccess 5.0:

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 Javadocs for ResponseBuilder for more info.

Using the ResponseBuilder#badRequestText method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.badRequestText(message).build();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance().contentLength(length).build();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
Response response = ResponseBuilder.continue100().build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance(HttpStatus.CONTINUE).build();
```

Using the ResponseBuilder#forbiddenText method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.forbiddenText().build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance(HttpStatus.FORBIDDEN)
                                   .contentType(CommonMediaTypes.TEXT_PLAIN)
                                   .body(HttpStatus.FORBIDDEN.getMessage())
                                   .build();
```

> **Note**
>
> This approach does not localize the response body. Use a TemplateRenderer instead.

Using the ResponseBuilder#forbiddenWithoutBody method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.forbiddenWithoutBody().build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance(HttpStatus.FORBIDDEN).build();
```

Before PingAccess 5.0:

```
Response response = ResponseBuilder.forbiddenWithoutBody(message).build();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
String message = ResponseBuilder.htmlMessage(caption, text);
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
Response response = ResponseBuilder.internalServerError(message).build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.internalServerError().body(message).build();
```

> **Note**
>
> This approach does not localize the response body. Use a TemplateRenderer instead.

Using the ResponseBuilder#internalServerErrorWithoutBody method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.internalServerErrorWithoutBody().build();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance().build()
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance(HttpStatus.INTERNAL_SERVER_ERROR).build();
```

Using the ResponseBuilder#noContent method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.noContent().build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance(HttpStatus.NO_CONTENT).build();
```

Using the ResponseBuilder#notFoundWithoutBody method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.notFoundWithoutBody().build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.notFound().build();
```

Using the ResponseBuilder#serverUnavailable method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.serverUnavailable(message).build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.serviceUnavailable().body(message).build();
```

> **Note**
>
> This approach does not localize the response body. Use a TemplateRenderer instead.

Using the ResponseBuilder#serviceUnavailableWithoutBody method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.serverUnavailableWithoutBody().build();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance().status(HttpStatus.OK).build();
```

After PingAccess 5.0:

```
Response response = ResponseBuilder.newInstance(HttpStatus.OK).build();
```

Using the ResponseBuilder#unauthorizedWithoutBody method

Before PingAccess 5.0:

```
Response response = ResponseBuilder.unauthorizedWithoutBody().build();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
boolean redirect = response.isRedirect();
```

After PingAccess 5.0:

```
boolean redirect = response.getStatusCode() >= 300
                   && response.getStatusCode() < 400;
```

Using the Response#setStatusCode method

Before PingAccess 5.0:

```
response.setStatusCode(HttpStatus.OK.getCode());
```

After PingAccess 5.0:

```
response.setStatus(HttpStatus.OK);
```

Using the Response#setStatusMessage method

Before PingAccess 5.0:

```
response.setStatusMessage(HttpStatus.OK.getMessage());
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
Date expiration = identity.getTokenExpiration();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
Date expiresAt = metadata.getExpiresAt();
```

After PingAccess 5.0:

```
Date expiresAt = Date.from(metadata.getExpiresAt());
```

Using the OAuthTokenMetadata#getRetrievedAt method

Before PingAccess 5.0:

```
Date retrievedAt = metadata.getRetrievedAt();
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
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 PingAccess 5.0:

```
private AccessExceptionContext createAccessExceptionContext(HttpStatus httpStatus,
                                                            Throwable cause)
{
    return AccessExceptionContext.create(httpStatus)
                                 .cause(cause)
                                 .exceptionMessage(httpStatus.getMessage())
                                 .errorDescription(httpStatus.getLocalizedMessage());
}
```

Before PingAccess 5.0:

```
private AccessExceptionContext createAccessExceptionContext(HttpStatus httpStatus,
                                                            String localizationKey,
                                                            String[] substitutions)
{
    return AccessExceptionContext.builder()
                                 .httpStatus(httpStatus)
                                 .errorDescription(localizationKey)
                                 .errorDescriptionIsKey(true)
                                 .errorDescriptionSubstitutions(substitutions)
                                 .build();
}
```

After PingAccess 5.0:

```
private AccessExceptionContext createAccessExceptionContext(HttpStatus httpStatus,
                                                            String localizationKey,
                                                            String[] substitutions)
{
    LocalizedMessage localizedMessage =
            new ParameterizedLocalizedMessage(localizationKey, substitutions);

    return AccessExceptionContext.create(httpStatus)
                                 .errorDescription(localizedMessage);
}
```

Before PingAccess 5.0:

```
private AccessExceptionContext createAccessExceptionContext(HttpStatus httpStatus,
                                                            String localizationKey)
{
    return AccessExceptionContext.builder()
                                 .httpStatus(httpStatus)
                                 .errorDescription(localizationKey)
                                 .errorDescriptionIsKey(true)
                                 .build();
}
```

After PingAccess 5.0:

```
private AccessExceptionContext createAccessExceptionContext(HttpStatus httpStatus,
                                                            String localizationKey)
{
    LocalizedMessage localizedMessage = new BasicLocalizedMessage(localizationKey);
    return AccessExceptionContext.create(httpStatus)
                                 .errorDescription(localizedMessage);
}
```

Before PingAccess 5.0:

```
private AccessExceptionContext createAccessExceptionContext(HttpStatus httpStatus)
{
    return AccessExceptionContext.builder()
                                 .from(httpStatus)
                                 .httpStatusDescription(httpStatus.getLocalizationKey())
                                 .httpStatusDescriptionIsKey(true)
                                 .templateFile("template.html")
                                 .contentType("text/html");
}
```

After PingAccess 5.0:

```
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 Javadocs 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 PingAccess 5.0:

```
private void throwAccessException(String errorDescription,
                                  Throwable throwable) throws AccessException
{
    throw new AccessException(errorDescription, throwable);
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void throwAccessException(String errorDescription) throws AccessException
{
    throw new AccessException(errorDescription);
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void createAccessException(int statusCode,
                                   String statusMessage,
                                   String errorDescription) throws AccessException
{
    throw new AccessException(statusCode, statusMessage, errorDescription);
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void throwAccessException(int statusCode,
                                  String statusMessage,
                                  String errorDescription,
                                  Throwable throwable) throws AccessException
{
    throw new AccessException(statusCode, statusMessage, errorDescription, throwable);
}
```

After PingAccess 5.0:

```
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 PingAccess 5.0:

```
private void throwAccessException() throws AccessException
{
    throw new AccessException(AccessExceptionContext.builder()
                                                    .httpStatusCode(403)
                                                    .httpStatusMessage("Forbidden")
                                                    .build());
}
```

After PingAccess 5.0:

```
private void throwAccessException() throws AccessException
{
    throw new AccessException(AccessExceptionContext.create(HttpStatus.FORBIDDEN));
}
```

Using the AccessException#getExceptionContext method

Before PingAccess 5.0:

```
AccessExceptionContext context = accessException.getExceptionContext();
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
int statusCode = accessException.getHttpStatusCode();
```

After PingAccess 5.0:

```
int statusCode = accessException.getErrorStatus().getCode();
```

Using the AccessException#getHttpStatusMessage method

Before PingAccess 5.0:

```
String statusMessage = accessException.getHttpStatusMessage();
```

After PingAccess 5.0:

```
String statusMessage = accessException.getErrorStatus().getMessage();
```

Using the AccessException#setHttpStatusCode method

Before PingAccess 5.0:

```
accessException.setHttpStatusCode(statusCode);
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
accessException.setHttpStatusMessage(statusMessage);
```

After PingAccess 5.0:

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 PingAccess 5.0:

```
@Override
public Outcome handleRequest(Exchange exchange) throws IOException
{
    Outcome outcome = applyPolicy(exchange);

    return outcome;
}
```

After PingAccess 5.0:

```
@Override
public Outcome handleRequest(Exchange exchange) throws AccessException
{
    Outcome outcome = applyPolicy(exchange);

    return outcome;
}
```

Account for the RuleInterceptor#handleResponse method signature change

Before PingAccess 5.0:

```
@Override
public void handleResponse(Exchange exchange) throws IOException
{
    applyPolicyToResponse(exchange.getResponse());
}
```

After PingAccess 5.0:

```
@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 PingAccess 5.0:

```
@Override
public Outcome handleRequest(Exchange exc)
        throws RuntimeException, IOException, InterruptedException
{
    // Site authenticator implementation //

    return Outcome.CONTINUE;
}
```

After PingAccess 5.0:

```
@Override
public void handleRequest(Exchange exc) throws AccessException
{
    // Site authenticator implementation //
}
```

Accounting for the SiteAuthenticatorInterceptor#handleResponse method signature chang

Before PingAccess 5.0:

```
@Override
public void handleResponse(Exchange exc) throws IOException
{
    // Site authenticator response implementation //
}
```

After PingAccess 5.0:

```
@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. 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.

Before PingAccess 5.0

```
private void invokeRenderResponse(TemplateRenderer templateRenderer,
                                  Map<String, String> context,
                                  String templateName,
                                  Exchange exchange,
                                  ResponseBuilder builder)
{
    templateRenderer.renderResponse(context, templateName, exchange, builder);
}
```

After PingAccess 5.0

```
private void invokeRenderResponse(TemplateRenderer templateRenderer,
                                  Map<String, String> context,
                                  String templateName,
                                  Exchange exchange,8
                                  ResponseBuilder builder)
{
    Response response = templateRenderer.renderResponse(exchange,
                                                        context,
                                                        templateName,
                                                        builder);
    exchange.setResponse(response);
}
```

iovation Device Risk Integration

The iovation Device Risk integration lets you supply data to iovation and allow or deny access based on an iovation result.

The iovation Device Risk integration must be installed on each node in the deployment. Once installed, it creates two new rules, which you can use to perform Device Risk checking and grant or deny access based on Device Risk's results.

The first rule, the iovation Device Risk Device Profiling rule, lets you gather data about the end user's system. This rule must be invoked before a request that uses the iovation Device Risk Authorization rule. It cannot be used for POST requests, and only functions on requests from a top-level browsing context.

The second rule, the iovation Device Risk Authorization rule, lets you allow or deny access based on Device Risk's evaluation of the user's system. This rule must be invoked after the iovation Device Risk Device Profiling rule, within the time period defined by the **Blackbox time to live (sec.)** field.

Both rules require authentication, so they can only be used on protected applications and resources. The rules are only applicable to Web applications.

Enable logging for iovation events by updating the `<PA_HOME>/conf/log4j2.xml` file.

You can improve the accessibility of iovation, even to users that block third-party content, by configuring a reverse proxy for communicating with iovation.

The following topics are discussed in this guide:

- Installing the iovation Device Risk integration

- Creating iovation Device Risk device profiling rules
- Creating iovation Device Risk authorization rules
- Logging iovation events
- Improving iovation accessibility using a reverse proxy

Installing the iovation Device Risk integration

You can install the iovation Device Risk integration in your environment to enable iovation Device Risk rules.

Steps

1. Download the iovation integration bundle from the PingAccess downloads page.
2. Stop PingAccess.
3. From the `.zip` file, copy the integration kit files to the PingAccess directory.
    1. Copy the `dist/pa-iovation-rules-version.jar` file to the `<PA_HOME>/deploy` directory.
    2. Copy the contents of the `dist/conf` directory to the `<PA_Home>/conf` directory.

    > **Note**
    > Preserve the relative location of each file within its subdirectory.

4. Edit the `iovation-messages.properties` file and copy its contents into the `pa-messages.properties` file.

    > **Note**
    > You can edit the values of the new properties as needed.

5. Start PingAccess.
6. If you operate PingAccess in a cluster, repeat steps 2-5 for each node.

Creating iovation Device Risk device profiling rules

Create a rule to perform iovation Device Risk device profiling on the end user's system.

Steps

1. Click **Access**, then go to **Rules > Rules**.
2. Click **+ Add Rule**.
3. In the **Name** field, enter a unique name. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. From the **Type** list, select **iovation Device Risk device profiling**.

5. In the **Blackbox cookie name prefix** field, enter the prefix of the cookies containing the iovation blackbox captured by this rule. The default value is `iovation_bb`.

   Enter the third-party service to use for fraud checks to iovation.

6. In the **iovation URI hook** field, enter the location from which to load the iovation blackbox collection JavaScript. This value can be a relative or absolute Uniform Resource Locator (URL) path, but cannot be a complete URL containing scheme and authority. The default value is `/iojs`.

   > **Note**
   >
   > To unconditionally disable the first-party iovation JavaScript, set this value to `<Reserved Application Context Root>/iojs`, where *<Reserved Application Context Root>* is the context root of the reserved PingAccess application (`/pa` by default).

7. **Optional:** If additional options need to be configured, click **Show Advanced**.

   1. In the **Blackbox time to live (sec.)** field, enter the number of seconds to use the iovation blackbox during a session before refreshing the blackbox device profile. The default value is `300`.

   2. **Optional:** In the **iovation subkey** field, enter the subkey value supplied to you by iovation.

      > **Note**
      >
      > This value is used for debugging and troubleshooting purposes.

   3. In the **iovation script version** field, enter the version of the iovation Device Recognition JavaScript to use. The default value is `general5`.

   4. **Optional:** Check the **Overwrite existing blackbox** checkbox to perform blackbox device profiling with every request to resources using this rule.

      When unchecked, the rule only collects the device profile after the existing blackbox expires. This option is unchecked by default.

8. Click **Save**.

Creating iovation Device Risk authorization rules

Create a rule to share device information with iovation Device Risk and allow or deny access based on the response.

About this task

When this rule runs, the iovation response is stored in the `com.pingidentity.pa.iovation.kit:policy.decision.outcome` property. Valid values are `allow`, `deny`, and `review`. This property can be used by Groovy rules or custom plugins.

Steps

1. Click **Access**, then go to **Rules > Rules**.

2. Click **+ Add Rule**.

3. In the **Name** field, enter a unique name. The name can be up to 64 characters long. Special characters and spaces are allowed.

4. From the **Type** list, select **Iovation Device Risk authorization**.
5. From the **iovation Service** list, select the third-party service to use for outbound fraud checks to iovation.
6. In the **Blackbox Cookie Name Prefix** field, enter the prefix of the cookies containing the iovation blackbox captured previously by the iovation Device Risk Device Profiling rule. The default value is `iovation_bb`.
7. In the **Subscriber ID** field, enter the subscriber ID provided to you by iovation.
8. In the **Subscriber Account** field, enter the subscriber account name provided to you by iovation.
9. In the **Subscriber Passcode** field, enter the passcode used to authorize your ID and account with iovation.
10. In the **iovation Integration Point** field, enter the integration point associated with the rule set you want to use.
11. **Optional:** In the **Account Code Attribute** field, enter the name of an attribute containing a unique identifier for each end-user to send to iovation as the account code.
12. **Optional:** In the **Transaction Insight Parameter Mappings** section, enter one or more mappings of identity attributes in PingAccess to iovation Transaction Insight Parameters. The attributes are provided to iovation in the specified parameters.
    1. In the **Attribute Name** field, enter the attribute to use as a source.
    2. In the **Transaction Insight Parameter** field, enter the iovation Transaction Insight Parameter to use for the specified attribute.
    3. **Optional:** Click **Add Row** to add one or more additional mappings.
13. If additional options need to be configured, click **Show Advanced**.

Advanced Option	Description
Fraud Check Frequency (ms)	The number of milliseconds between iovation fraud checks. The default value is `20000`.
iovation Fraud Check API Endpoint	The application programming interface (API) endpoint to which iovation fraud check requests are directed. If not specified, a value of `/fraud/v1/subs/subscriberId/checks` is used, where *subscriberId* is the value in the **Subscriber ID** field.
iovation Failure Mode	Specifies whether PingAccess should allow or deny access if the communication with iovation is not completed successfully. The default value is **Deny**.

Advanced Option	Description
Invalid Blackbox Failure Mode	Specifies whether PingAccess should allow or deny access if the blackbox device profile is not in a usable state. This situation can occur when the blackbox has not already been collected from a previous exchange processed by this rule or when the collected blackbox has reached the end of its lifetime. The default value is **Deny**, which denies access. A value of **Continue** performs a risk assessment with no blackbox profile, while a value of **Allow** allows access.
iovation Protocol Error Handling	This section specifies the error parameters to use on a failure if there is a failure to communicate with iovation for the fraud check API request. In the **Error Response Code** field, enter the HTTP response code for the error response. In the **Error Response Template File** field, you can enter the name of a customized error page template if you do not want to use the default error page. Templates are stored in the `<PA_HOME>/conf/template/` directory. In the **Error Response Content Type** field, you can specify the content type if you are using a custom error response template file.
Review Fallback Type	Specifies whether PingAccess should allow or deny access if iovation returns a review result from the risk assessment. The default value is **Deny**.
Review Deny Handling	This section specifies the error parameters to use on a failure if the **Review Fallback Type** is set to **Deny**. In the **Error Response Code** field, enter the HTTP response code for the error response. In the **Error Response Template File** field, you can enter the name of a customized error page template if you do not want to use the default error page. Templates are stored in the `<PA_HOME>/conf/template/` directory. In the **Error Response Content Type** field, you can specify the content type if you are using a custom error response template file.

Advanced Option	Description
Deny Handling	This section specifies the error parameters to use on a failure if iovation returns a Deny (D) result or when the blackbox is not set and the **Invalid Blackbox Failure Mode** is set to Deny. In the **Error Response Code** field, enter the HTTP response code for the error response. In the **Error Response Template File** field, you can enter the name of a customized error page template if you do not want to use the default error page. Templates are stored in the `<PA_HOME>/conf/template/` directory. In the **Error Response Content Type** field, you can specify the content type if you are using a custom error response template file.

14. Click **Save**.

Logging iovation events

Update the PingAccess logging file to log iovation events.

About this task

This procedure modifies the existing `<PA_HOME>/conf/log4j2.xml` file to log communications with iovation to a new log file. In a clustered environment, you must perform this procedure on every node.

Steps

1. Edit the `<PA_HOME>/conf/log4j2.xml` file.
2. Locate the Appenders section and add a section to create the new log file.

   Example:

```
<RollingFile name="Iovation-File"
                     fileName="${sys:pa.home}/log/pingaccess_iovation_audit.log"
                     filePattern="${sys:pa.home}/log/pingaccess_iovation_audit.%d{yyyy-MM-dd}.log"
                     ignoreExceptions="false">
            <PatternLayout>
                <pattern>%d{ISO8601}| %X{exchangeId}| %X{IOVATION_AUDIT.trackingNumber} |
%X{IOVATION_AUDIT.deviceAlias} | %X{IOVATION_AUDIT.accountCode} | %X{IOVATION_AUDIT.result} |
%X{IOVATION_AUDIT.reason} | %X{IOVATION_AUDIT.ruleName} | %X{IOVATION_AUDIT.iovationId} |
%X{IOVATION_AUDIT.statedIp} %n</pattern>
            </PatternLayout>
            <Policies>
                <TimeBasedTriggeringPolicy/>
            </Policies>
</RollingFile>
```

   This example uses a log file name of `<PA_HOME>/log/pingaccess_iovation_audit.log`.

The following variables are used in this example.

Variable	Definition
`%d`	The transaction time.
`exchangeId`	The ID for a specific request/response pair.
`IOVATION_AUDIT.trackingNumber`	An iovation-assigned unique ID for the transaction that can be used to locate the transaction in searches and reports.
`IOVATION_AUDIT.deviceAlias`	The iovation identifier for the requesting device. If no blackbox is present at the time of the iovation authorization request, a value of 0 is used.
`IOVATION_AUDIT.accountCode`	The value of the `accountCode` attribute for the transaction.
`IOVATION_AUDIT.result`	The iovation risk check result. Valid values are: ◦ A – Accept ◦ D – Deny ◦ R – Review
`IOVATION_AUDIT.reason`	The iovation admin-specified value corresponding to the iovation rule that contributed most to the result.
`IOVATION_AUDIT.ruleName`	The name of the PingAccess rule responsible for this iovation Fraud check.
`IOVATION_AUDIT.iovationId`	A unique ID provided by iovation for the request.
`IOVATION_AUDIT.statedIp`	The Internet Protocol (IP) address of the requesting client. This value is provided as the statedIp of the iovation Fraud API request.

3. Locate the Loggers section and add an entry to enable logging.

   Example:

   ```
   <Logger name="iovationaudit" level="INFO" additivity="false">
         <AppenderRef ref="Iovation-File"/>
   </Logger>
   ```

4. Restart PingAccess.

Improving iovation accessibility using a reverse proxy

You can improve the accessibility of iovation, even if end-users are blocking third-party content, by configuring a reverse proxy to connect to iovation.

About this task

This procedure applies only for gateway deployments. It automates much of the process of configuring a reverse proxy to connect to iovation, such that PingAccess acts as a reverse proxy as described by Retrieving & Serving Dynamic iovation JavaScript in the iovation Help Center. It uses Postman to create a PingAccess application with settings that allow it to act as the reverse proxy for the provided virtual hosts. You can add additional virtual hosts to this application as necessary.

If you are using an agent deployment, iovation recommends one of two models: Reverse proxy or Web device print server. See Retrieving & Serving Dynamic iovation JavaScript for more information about these models.

Steps

1. Go to the `<iovation_integration_home>/setup` directory.
2. Import the `iovation First Party Dynamic JavaScript Reverse Proxy.postman_collection.json` collection file into Postman.
3. Set the environment variables as described by the collection documentation.
4. Run the collection.

IWA Integration

Integrated Windows Authentication (IWA) is a process that allows users to authenticate with Windows credentials using the Kerberos or (legacy) NTLM protocols.

Unlike session-based authentication, IWA relies on authenticating client-server connections, which are then given access to protected content. PingAccess handles these connections differently, although configuration in the Admin UI is identical to normal applications. This document is intended to clarify IWA connection handling in PingAccess and help administrators avoid common mistakes in this configuration.

> **Note**
>
> For IWA to work, every node in the network architecture must support bound connections, including load balancers, gateways, and proxies. If a network component in front of PingAccess improperly re-uses an authenticated connection, PingAccess might break this connection to prevent session stealing.
> The AWS ELB does not support IWA.
> NTLM is no longer supported in PingFederate, however NTLM connections are treated the same as Kerberos connections in PingAccess.

Setting up IWA using PingFederate

About this task

Set up an application to be protected with Kerberos authentication using PingFederate's Kerberos Adapter, while PingFederate is protected by PingAccess:

Steps

1. Configure your Kerberos adapter in PingFederate.

   For more information, see Configure a Kerberos adapter instance in the PingFederate documentation.

2. Add a new site in PingAccess.

   1. Go to **Applications → Sites** and click **+ Add Site**.
   2. In the **Name** field, enter a desired name for the site.
   3. In the **Targets** field, enter one or more hostname:port pairs for the site.

      The host and port to point to PingFederate on port `9031`.
   4. Click **Save**.

   For more information, see Adding sites.

3. Add a new application in PingAccess.

   1. Go to **Applications → Applications** and click **+ Add Application**.
   2. In the **Name** field, enter a desired name for the site.
   3. In the **Context Root** field, specify the first part of the URL path for the application and its resources.
   4. In the **Virtual Host** field, enter the host desired for the target application.
   5. In the **Destination** list, select **Site**.
   6. In the **Site** list, select the PingFederate site previously created.
   7. Configure the remaining fields as desired. Click **Save**.

   For more information, see Adding an application.

4. Enable the application.

   Result:

   The protected application can utilize the Kerberos protocol for authentication through PingAccess, using PingFederate.

Setting up IWA directly

About this task

Set up PingAccess to manage an application that already uses IWA for authentication.

Steps

1. Add a new site in PingAccess.

   1. Go to **Applications → Sites** and click **+ Add Site**.
   2. In the **Name** field, enter a desired name for the site.

3. In the **Targets** field, enter one or more hostname:port pairs for the site.
4. Click **Save**.

For more information, see Adding sites.

2. Add a new Application in PingAccess.
    1. Go to **Applications → Applications** and click **+ Add Application**.
    2. In the **Name** field, enter a desired name for the site.
    3. In the **Context Root** field, specify the first part of the URL path for the application and its resources.
    4. In the **Virtual Host** field, enter the host desired for the target application.
    5. In the **Destination** list, select **Site**.
    6. In the **Site** list, select the site for this application.
    7. Configure the remaining fields as desired. Click **Save**.

    For more information, see Adding an application.
3. Enable the application.

    Result:

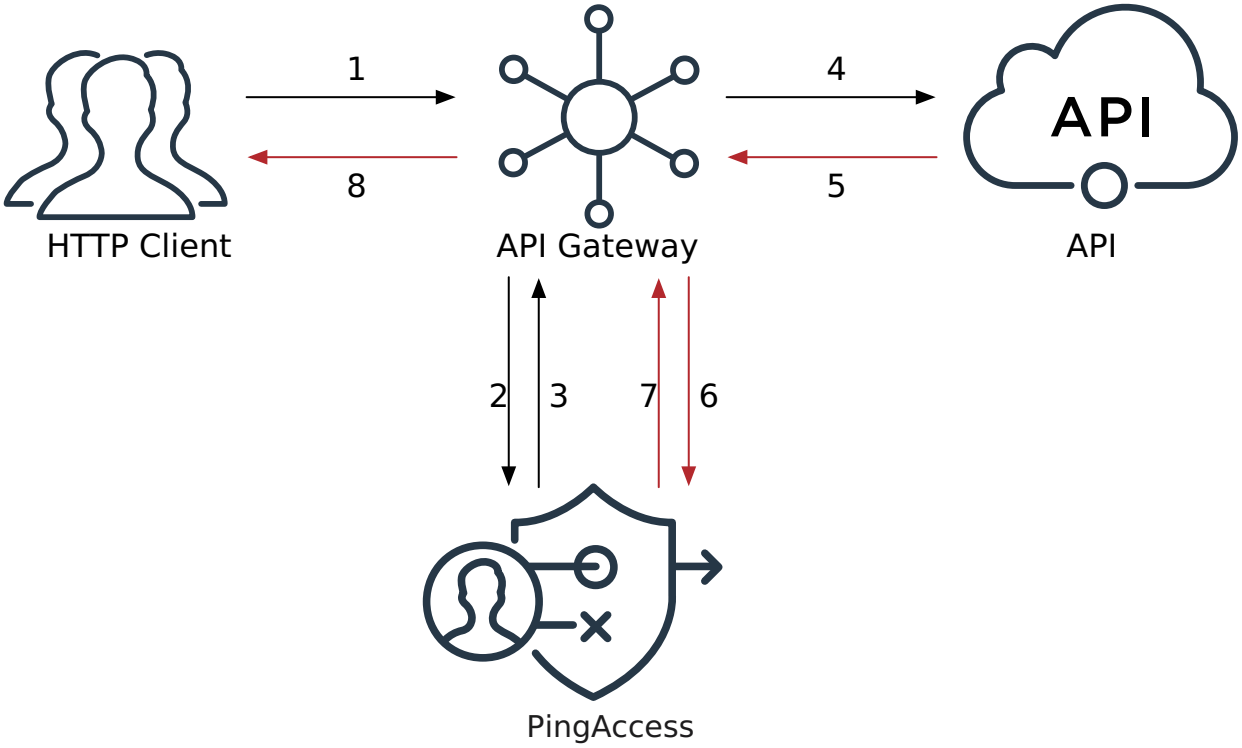
    The protected application can utilize the Kerberos protocol for authentication through PingAccess.

Kong API Gateway Integration

Ping Identity provides a Kong Gateway integration that enables the use of PingAccess and other Ping Identity products for policy decisions.

Integration with Kong Gateway allows PingAccess to handle the complexities of the OAuth and OpenID Connect (OIDC) protocols, making it easier to manage access control in your API. Instead of making access control configurations repeatedly, install and configure the Kong plugin once and manage your access control rules in PingAccess.

The following diagram explains how traffic flows through Kong Gateway and PingAccess.



1. The HTTP client sends an inbound request to the API gateway.
2. The API gateway sends a sideband request to PingAccess.
3. PingAccess evaluates the request and sends a response to the API gateway.
4. The API gateway analyzes the response from PingAccess to determine whether the request should be forwarded to the API and, if so, whether any modifications should be made to the request.

   If the request is denied, PingAccess includes directives to influence how the API gateway responds to the HTTP Client.
5. The API sends an outbound response to the API gateway.
6. The API gateway passes the response to PingAccess for processing.
7. PingAccess sends a response to the API gateway.
8. The API gateway processes the response from PingAccess.

   If modifications should be made, the response to the HTTP client includes directives for modifying the response.

Important

Review the following usage considerations before setting up the Kong plugin:

Mutual TLS (mTLS)

This plugin supports client certificate authentication using mTLS. However, this feature requires using the `mtls-auth` plugin (only available in the Enterprise edition of Kong) in conjunction with `ping-auth`. Learn more in the Kong mTLS-auth documentation.

When configured, the `mtls-auth` plugin uses the mTLS process to retrieve the client certificate, which allows `ping-auth` to provide the certificate in the `client_certificate` field of the sideband requests.

Transfer-encoding

Because of an outstanding defect in Kong, `ping-auth` is unable to support the `Transfer-Encoding` header, regardless of the value.

Logging limit

Because of OpenResty's log level limit, log messages are limited to 2048 bytes by default, which is less than the size of many requests and responses. Learn more in the OpenResty reference documentation.

HTTP/2

The Kong Gateway integration does not support HTTP/2.

To set up the Kong Gateway integration:

1. Configure a sideband client in PingAccess to create a shared secret for Kong Gateway.
2. Configure the `ping-auth` plugin in Kong Manager.
3. Create a PingAccess application for the protected API and verify the connection between PingAccess and Kong Gateway.

Configuring PingAccess for Kong Gateway integration

To allow Kong Gateway to use the PingAccess Sideband API for authorization and request and response modification, create a sideband client in PingAccess that represents Kong Gateway.

Before you begin

- Verify that the `sideband.http.enabled` property is set to `true` in the PingAccess `run.properties` file. Learn more in the Configuration file reference.
- Review the **Sideband model** section in How do I choose a deployment model? and Protecting an API with PingAccess in a sideband deployment.

About this task

Prepare PingAccess to authenticate requests from Kong Gateway by configuring a header name and shared secret for Kong Gateway to authenticate with.

Note

You can find more information about sideband clients in Sideband Clients.

Steps

1. In the PingAccess admin console, click **Applications**, then go to **Sideband Clients**.
2. Click **+ Add Sideband Clients**.
3. In the **Name** field, enter a unique name for the sideband client.
4. (Optional) In the **Description** field, enter a description for the sideband client.
5. In the **Secrets** section, add a secret:

   Result:

   The **New Secret** page opens.
6. On the **New Secret** page, click **Copy** to copy the new secret to your clipboard.

   > **Important**
   > - Save the secret in a secure location. You'll use it in the following procedure.
   > - Version 1.2.0 of the `ping-auth` plugin supports referenceable secrets. For security reasons, store the shared secret in a vault supported by Kong. Learn more in Secrets Management and Environment Variables Vault in the Kong documentation.
   > - When you use the secret, replace the `<client name>` placeholder value with the **Name** you configured in step 3.
7. Click **Done**.
8. (Optional) In the **Header Name** field, enter a unique, descriptive name to make it easier to tell your configured secrets apart.
9. Click **Save**.

Next steps

Configure Kong Gateway for integration with PingAccess.

Configuring Kong Gateway for PingAccess integration

Download, install, and configure the `ping-auth` plugin to set up a connection between Kong Gateway and PingAccess.

Before you begin

Install and start Kong Gateway. Learn more in the Kong Gateway documentation.

Steps

1. Download and extract the `ping-auth` plugin for Kong Gateway from https://luarocks.org/modules/pingidentity/kong-plugin-ping-auth.
2. Install the `ping-auth` plugin using one of the following procedures:
   - **Via LuaRocks from the created 'rock'**: If you use this procedure, run the `luarocks install kong-plugin-ping-auth` command to install the plugin.

- **Manually**: Learn more in Kong's installation guide.

3. After installation, load the plugin into Kong:

    1. Edit the `plugins` property in the `kong.conf` file to include the `ping-auth` plugin.

        Example:

        ```
        plugins = bundled,ping-auth
        ```

    2. Restart Kong Gateway to apply your changes.

    3. To confirm that Kong loaded the plugin successfully, look for the debug-level message `Loading plugin: ping-auth` in Kong's `error.log` file.

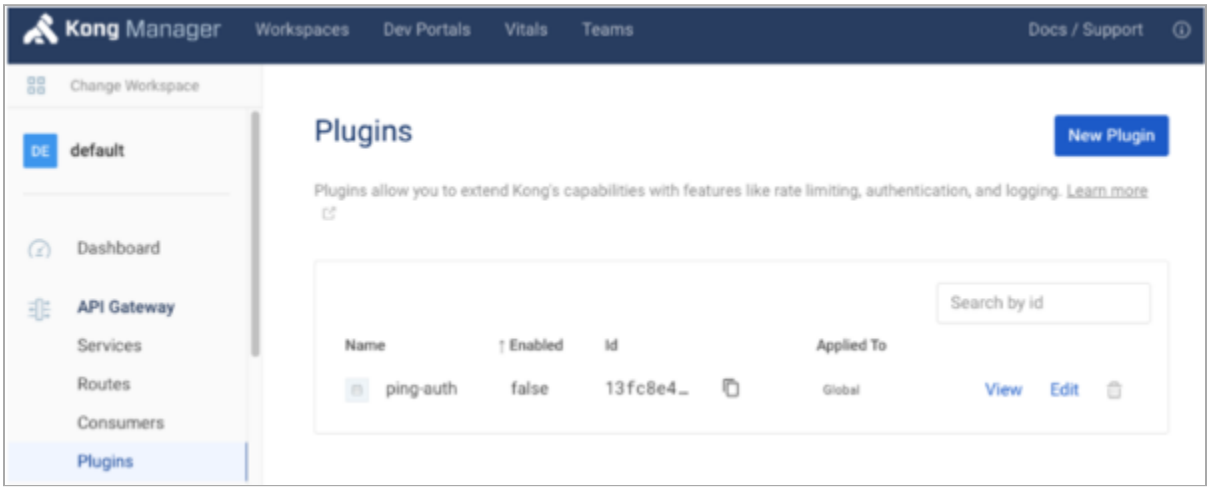
    You can find more information and configuration tips in Load the plugin and Verify loading the plugin in the Kong Gateway documentation.

4. Use the Kong Gateway UI or API to complete the configuration.

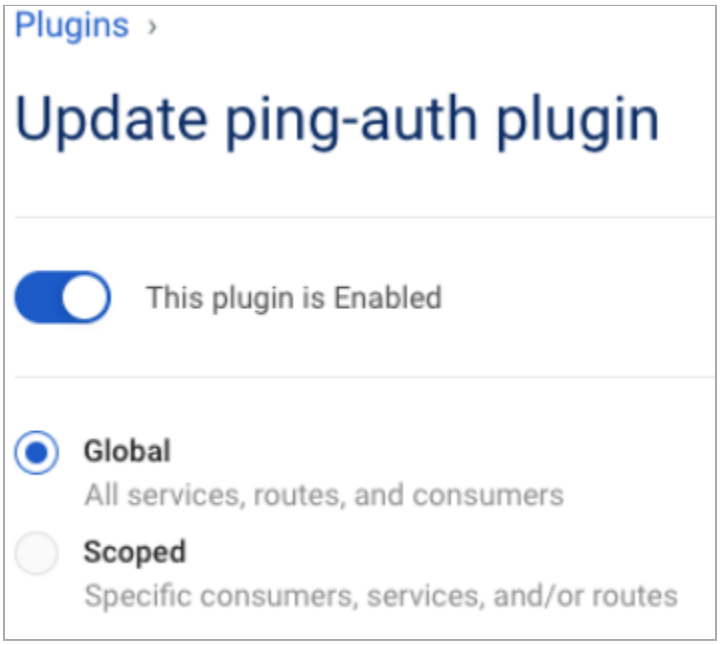
Kong Gateway UI

Setting up Kong Gateway
Steps

1. In Kong Manager, select the **Default** workspace, then click **Plugins**.



2. On the `ping-auth` line, click **Edit**
3. On the **Update ping-auth plugin** page, click the toggle to enable the plugin.

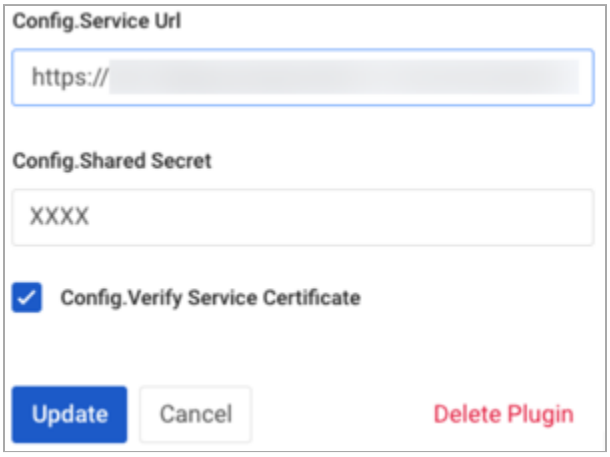


4. (Optional) If you want to enable the plugin for specific consumers, services, or routes, click **Scoped** and then enter **Service**, **Route**, and **Consumer** information as needed.
5. In the **Config.Service URL** field, enter the full URL for PingAccess, using the `https://<PINGACCESS_URL>:<SIDEBAND_PORT>/` format.

   Example:

   https://localhost:3020/

> **Note**
>
> - `<PINGACCESS_URL>` is the fully-qualified name of the machine running PingAccess.
> - Don't include `/sideband…` in the path.
> - The default sideband port is `3020`, but you should check the `sideband.http.port` property in the PingAccess `run.properties` file to confirm that you haven't configured a different value. Learn more in the Configuration file reference.
> - The `sideband.http.enabled` property must be set to `true` in the PingAccess `run.properties` file for the `ping-auth` plugin to communicate with PingAccess successfully.



6. In the **Config.Shared Secret** field, paste the shared secret you created in Configuring PingAccess for Kong Gateway integration.

> **Important**
>
> Version 1.2.0 of the `ping-auth` plugin supports referenceable secrets. For security reasons, store the shared secret in a vault supported by Kong. Learn more in Secrets Management and Environment Variables Vault in the Kong documentation.

7. In the **Config.Secret Header Name** field, enter the name of the header that provides the shared secret.

   The default value is `CLIENT-TOKEN`.

8. (Optional) Configure additional options in Kong Manager or the API:

> **Note**
>
> You can find information on using the API to configure these fields in Create a plugin in the Kong Gateway admin API documentation.

Option	API Field Name	Description
`Config.Connection KeepAliveMs`	`connection_keepAlive_ms`	The duration to keep the connection alive for reuse. The default value is `6000`.

Option	API Field Name	Description
`Config.Connection Timeout Ms`	`connection_timeout_ms`	The duration to wait before the connection times out. The default value is `10000`.
`Config.Enable Debug Logging`	`enable_debug_logging`	Controls if requests and responses are logged at the debug level. The default value is `false`. **Note** For log messages to show in the `error.log`, you must set `log_level = debug` in the `kong.conf` file.
`Config.Verify Service Certificate`	`verify_service_certificate`	Controls whether the service certificate is verified. This configuration option is intended for testing purposes. The default value is `true`.

9. Click **Update**, then click **Update Plugin**.

Next steps

Create a PingAccess application for the protected API and verify the connection between PingAccess and Kong Gateway.

Kong Gateway API

Setting up Kong Gateway using the API
Steps

1. Include the following JSON object in a POST request to `https://<KONG_URL>/plugins`:

```
{
    "name": "ping-auth",
    "enabled": true,
    "config": {
        "service_url": "https://<PINGACCESS_URL>:3020/",
        "shared_secret": "<SHARED_SECRET>",
        "secret_header_name": "<HEADER_NAME>"
    }
}
```

   Use the following information to fill out the required fields in the **Config** section:

   - `service_url`: The full URL of the Ping policy provider, using the `https://<PINGACCESS_URL>:<SIDEBAND_PORT>/` format. The default value is `https://localhost:3020/`.

     Note

     - `<PINGACCESS_URL>` is the fully-qualified name of the machine running PingAccess.
     - Don't include `/sideband…` in the path.
     - The default sideband port is `3020`, but you should check the `sideband.http.port` property in the PingAccess `run.properties` file to confirm that you haven't configured a different value. Learn more in the Configuration file reference.
     - The `sideband.http.enabled` property must be set to `true` in the PingAccess `run.properties` file for the `ping-auth` plugin to communicate with PingAccess successfully.

   - `shared_secret`: The shared secret used to authenticate this plugin to the policy provider. Paste the shared secret you created in Configuring PingAccess for Kong Gateway integration.

     Important

     Version 1.2.0 of the `ping-auth` plugin supports referenceable secrets. For security reasons, store the shared secret in a vault supported by Kong. Learn more in Secrets Management and Environment Variables Vault in the Kong documentation.

   - `secret_header_name`: The name of the header that provides the shared secret. The default value is `CLIENT-TOKEN`.

2. (Optional) Include additional fields in the POST request, making sure to adhere to the Kong API specification.

   Learn more in Create a plugin in the Kong Gateway admin API documentation.

Option	API Field Name	Description
`Config.Connection KeepAlive Ms`	`connection_keepAlive_ms`	The duration to keep the connection alive for reuse. The default value is `6000`.
`Config.Connection Timeout Ms`	`connection_timeout_ms`	The duration to wait before the connection times out. The default value is `10000`.
`Config.Enable Debug Logging`	`enable_debug_logging`	Controls if requests and responses are logged at the debug level. The default value is `false`. **Note** For log messages to show in the `error.log`, you must set `log_level = debug` in the `kong.conf` file.
`Config.Verify Service Certificate`	`verify_service_certificate`	Controls whether the service certificate is verified. This configuration option is intended for testing purposes. The default value is `true`.

Next steps

Create a PingAccess application for the protected API and verify the connection between PingAccess and Kong Gateway.

Verifying the connection

About this task

You can verify the connection between PingAccess and Kong Gateway by making an end-to-end request. To do so, first create an application for Kong in PingAccess.

Note

This procedure contains only information that's specific to creating a working Kong Gateway connection. Learn more about configuration settings outside the scope of this procedure in Adding an application and Application field descriptions.

Steps

1. In the PingAccess admin console, go to **Applications > Applications** and click **+ Add Application**.

2. In the **Name**, **Context Root**, and **Virtual Host(s)** fields, enter information matching how the application APIs are exposed.
3. In the **Application Type** list, select **API**.
4. In the **Destination** list, select **Sideband**.
5. In the **Sideband Client** list, select the sideband client you created for Kong in Configuring PingAccess for Kong Gateway integration.
6. Select the **Enabled** checkbox to allow the application to process requests.
7. After you've finished filling out the configuration, click **Save**.

Troubleshooting:

- If the requests aren't going through to the backend service as expected, debug logging can give a detailed breakdown of the request/response flow through Kong. To enable Kong debug logging:
    1. In the `ping-auth` config, set `enable_debug_logging` to true.
    2. In the `kong.conf` file, set `log_level = debug`.

    > **Note**
    >
    > These messages might contain sensitive request information and accumulate disk space, so you should disable debug logging after troubleshooting.

- The `pingaccess.log` on the PingAccess engine node can also provide useful debugging data. You can find information about how to access and interpret the PingAccess logs in Log configuration.

Next steps

After you create the application, you can assign rules to it to control access to APIs protected by the Kong gateway. Learn more about configuring rules in Rule management.

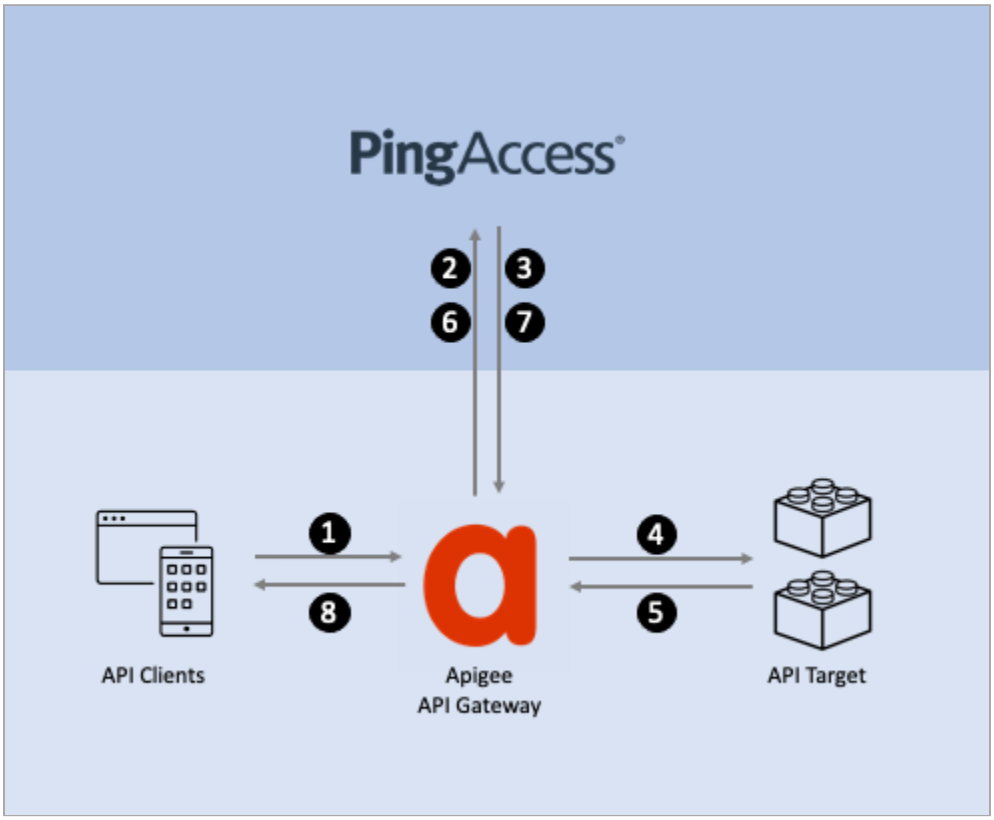
Apigee API Gateway Integration

Ping Identity's shared flow for Apigee extends Apigee's authorization capabilities through an external authorization policy runtime service.

Integration with Apigee allows identity and access management (IAM) administrators to centrally manage access control and application protection in PingAccess, while enforcement is delegated to Apigee.

Use this guide to install and configure the shared flow in Apigee. After installation and configuration, you can manage access control rules, identity mappings, and other application protection features in PingAccess.

The following diagram shows how traffic flows through Apigee and PingAccess.



1. The API client makes a request to the application programming interface (API) gateway.
2. The shared flow extracts fields from the API client's request and sends them to PingAccess for authorization.
3. PingAccess evaluates the request, validates the authorization, then responds to Apigee. The response could be an authentication or authorization error that should be immediately sent back to the client, or it could be a modified request that Apigee will send to the API target.
4. If authorized to proceed, Apigee passes the original or modified API request to the API target.
5. The API service responds with the requested resource or with the result of the operation.
6. The shared flow extracts fields from the API target's response and sends them to PingAccess for processing.
7. PingAccess responds to the processing request. The API response can be modified by the web session configuration and processing rules in PingAccess.
8. Apigee responds to the API client with the original API response received from the API target or the modified response received from PingAccess.

Before you begin

The Ping Identity shared flow for Apigee supports Apigee Edge, Apigee Private Cloud, and Apigee X. Before you begin, ensure that you have the following:

- A supported Apigee environment
- PingAccess installed and started

- The PingAuth shared flow bundle `.zip` file (`sharedflowbundle.zip`)

Configuring PingAccess for Apigee integration

About this task

Before Apigee can use PingAccess as an external authorization policy runtime service, you must prepare PingAccess to receive authorization requests from Apigee.

Steps

1. Enable the Sideband service:
    1. Edit the `<PA Home>/conf/run.properties` file and set `sideband.http.enabled=true` .
    2. **Optional:** By default, PingAccess will listen for sideband clients on port 3020. You can choose a different port by editing the value of the `sideband.http.port` property.
    3. Restart PingAccess.
2. Add a sideband client for Apigee:
    1. Go to **Applications → Sideband Clients** and click **+Add Sideband Client**.
    2. Give the client a name that helps you identify the Apigee environment, such as `Apigee-dev` .
    3. Click **+Add Secret**.
    4. Keep the header name of `CLIENT-TOKEN` unchanged, and copy the shared secret value.

       You will need this during the Apigee configuration.
    5. Click **Save**.
3. **Optional:** Download the sideband listener HTTPS certificate.

   By default, the PingAuth shared flow is configured to only trust the PingAccess Sideband Listener HTTPS certificate if it is issued from a well-known certificate authority (CA). To trust specific HTTPS certificates for PingAccess servers:
    1. Go to **Security → Key Pairs**.
    2. Click the **Pencil** icon next to the key pair labeled **SIDEBAND**.
    3. Click **Download Certificate** and save the public key certificate. You will need this during the Apigee configuration.

Configuring Apigee for PingAccess integration

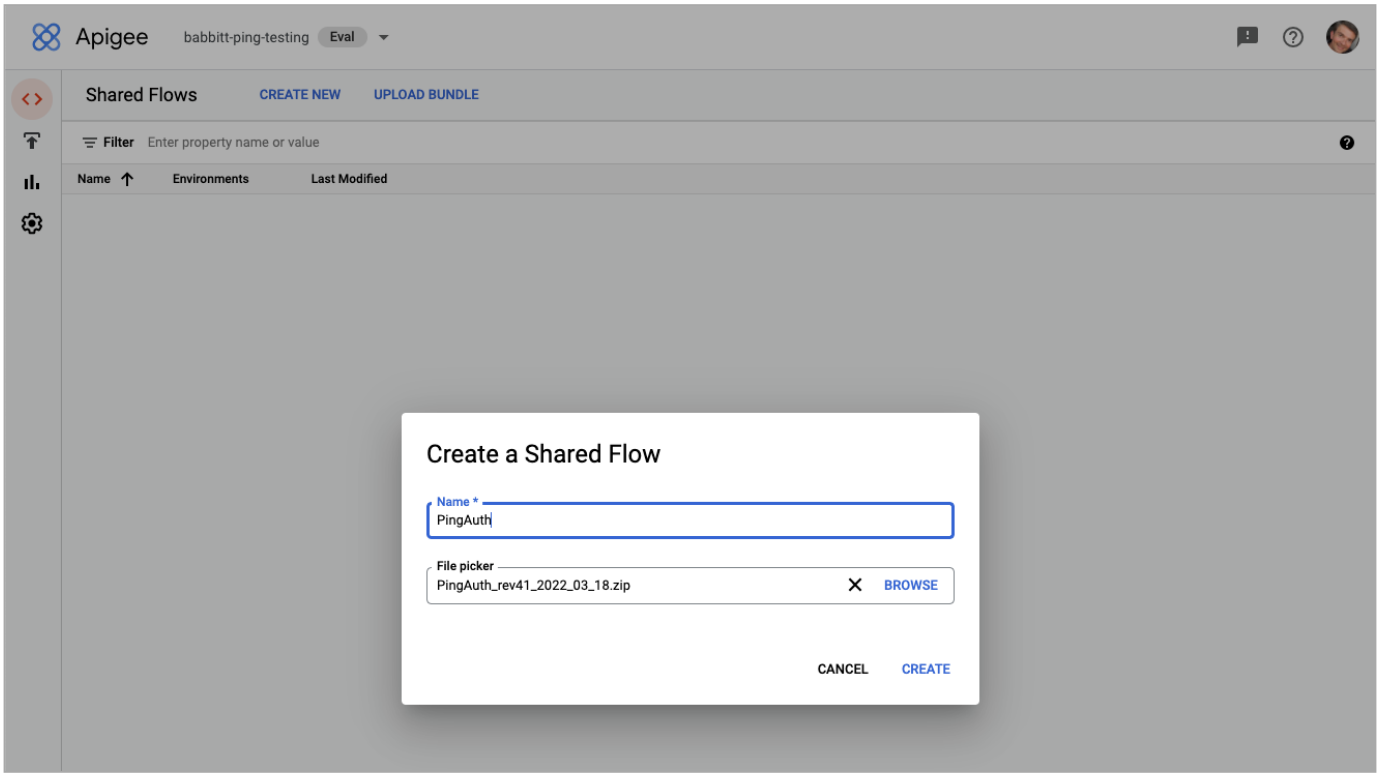
Install the PingAuth shared flow bundle in Apigee and configure it to integrate with PingAccess.

Steps

1. Upload the shared flow bundle.

Choose from:

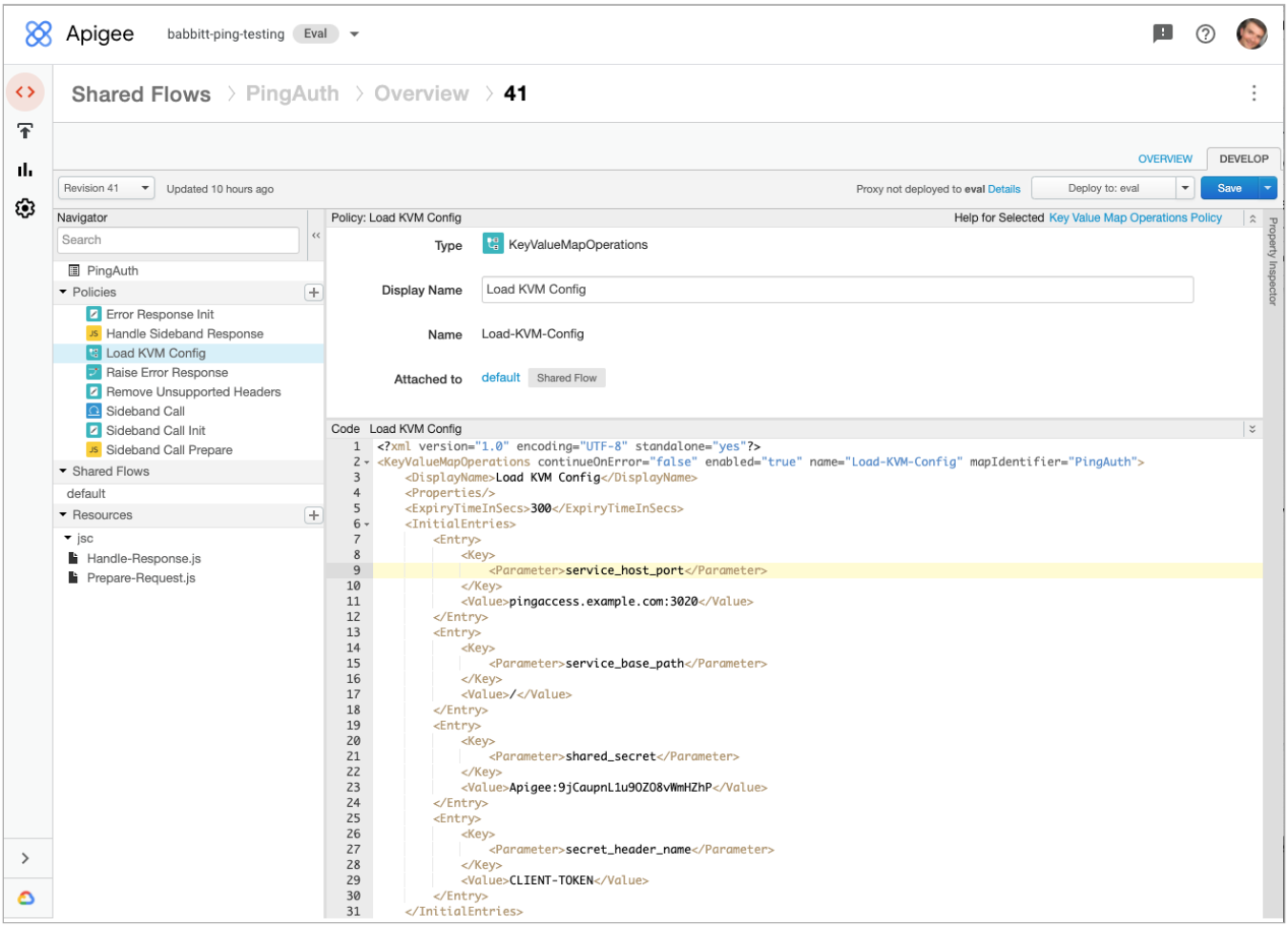
- If you are using Apigee X, go to **Develop → Shared Flows** and click **Upload Bundle**. Upload the PingAuth shared flow `.zip` file, and name the shared flow `PingAuth`.
- If you are using Apigee Edge or Apigee Private Cloud, click **+Shared Flow**, then click **Upload Bundle**. Upload the PingAuth shared flow bundle `.zip` file, and name the shared flow `PingAuth`.



2. If you are using Apigee X, configure the connection to PingAccess.

> **Note**
>
> Unlike Apigee Edge, Apigee X doesn't currently support managing the configuration values stored in key value maps in the Apigee interface. You must add these configuration values to the key value map policy. The key value map is created and the configuration values are added the first time the PingAuth shared flow is executed at runtime.

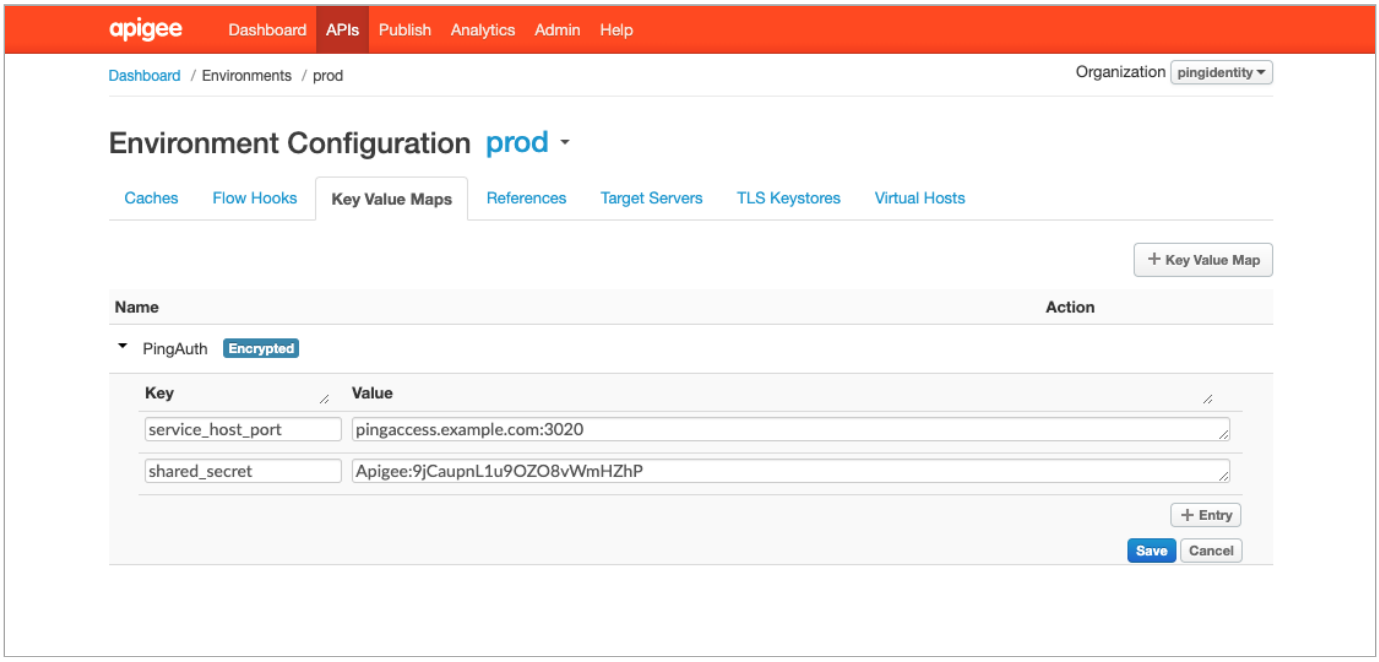
1. Go to the PingAuth shared flow at **Develop → Shared Flows → PingAuth**.
2. Click the **Develop** tab and examine **Revisions** to make sure you are on the latest revision.
3. In the **Policies** panel on the left, click the **Load KVM Config** policy.
4. In the policy editor panel, remove the comment lines above and below the `InitialEntries` element.
5. Edit the values for `service_host_port` and `shared_secret` to match the values that you obtained earlier.
6. Click **Save**.



3. If you are using Apigee Edge or Apigee Private Cloud, configure the connection to PingAccess.

   Apigee Edge stores environment-specific configuration values in key value maps so that the same policies can be used across multiple deployment environments without any changes to the policies.

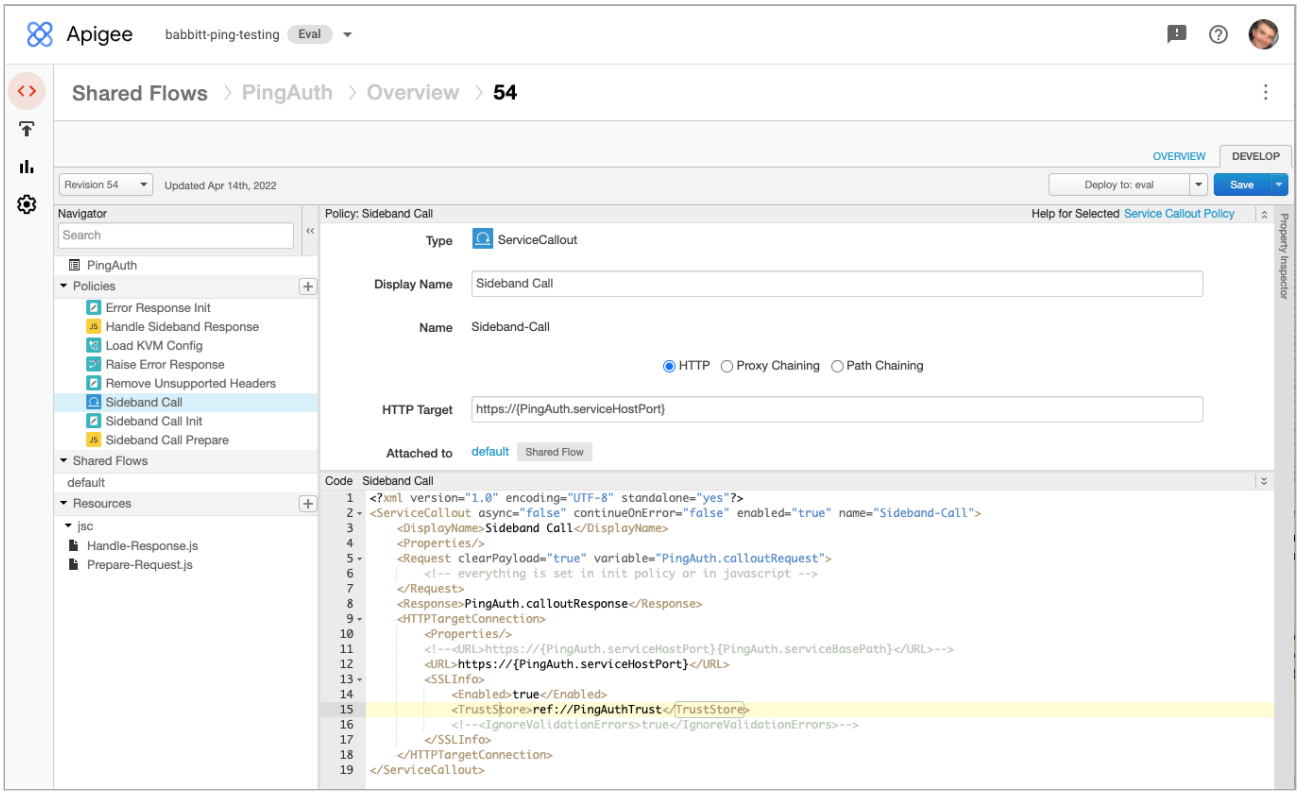
   1. Go to **Environment → Key Value Maps** and click **+Key Value Map**.
   2. Edit the key value map and click **Add Entry**. Use the key names `service_host_port` and `shared_secret`, and set the values to match the ones that you obtained earlier.
   3. Click **Save**.



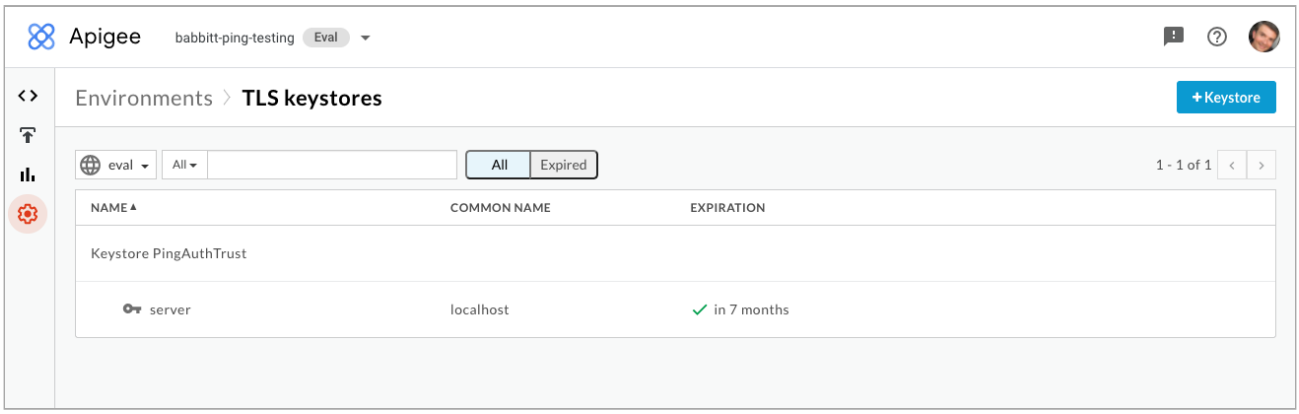
4. **Optional:** Configure HTTPS trust for PingAccess.

   By default, the PingAuth shared flow is configured to only trust the PingAccess Sideband Listener HTTPS certificates if it is issued from a well-known CA. To trust specific HTTPS certificates for PingAccess servers:

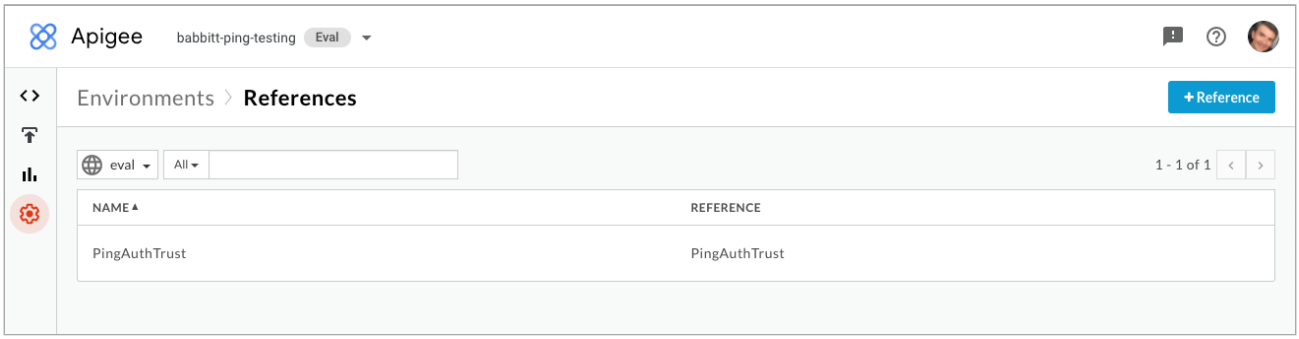
   1. Go to the PingAuth shared flow at **Develop → Shared Flows → PingAuth**.
   2. Click the **Develop** tab and examine **Revisions** to make sure you are on the latest revision.
   3. In the **Policies** section of the **Navigator**, click the **Sideband Call** policy.
   4. In the policy editor panel, remove the comment characters surrounding the `TrustStore` element.



5. Click **Save**.
6. Go to **Environment → TLS Keystores** and click **+Keystore**.
7. Give the key store a name that helps you identify your PingAccess environment, such as `PingAccess-dev-truststore`.
8. Click **+** to add a certificate, give the certificate an alias, and upload the certificate that you obtained earlier. Click **Save**.



9. Go to **Environment → References** and click **+Reference**.
10. In the **Name** field, enter `PingAuthTrust`.
11. Select the key store you created earlier, then click **Save**.



5. Deploy the shared flow:
    1. Go to **Develop → Shared Flows → PingAuth**.
    2. Deploy the most recent revision of the shared flow to your environment.

Adding an API Proxy in Apigee

Configure the API Proxy where you want to reach a target endpoint.

Steps

1. Go to **API Proxies → Create Proxy** and click the **Reverse Proxy** tab.
2. In the **Name** field, enter a name for the API proxy.
3. In the **Base Path** field, enter the base path of the API.
4. In the **Target (existing API)** field, enter the Uniform Resource Locator (URL) of the service invoked by the proxy.
5. Click **Next**.
6. In the **Common Policies** section, select **Pass through (no authorization)**.
7. Click **Next**.
8. In the **Optional Deployment** section, select the deployment environment.
9. Click **Create and Deploy**.

Attaching the PingAuth shared flow to API proxies in Apigee

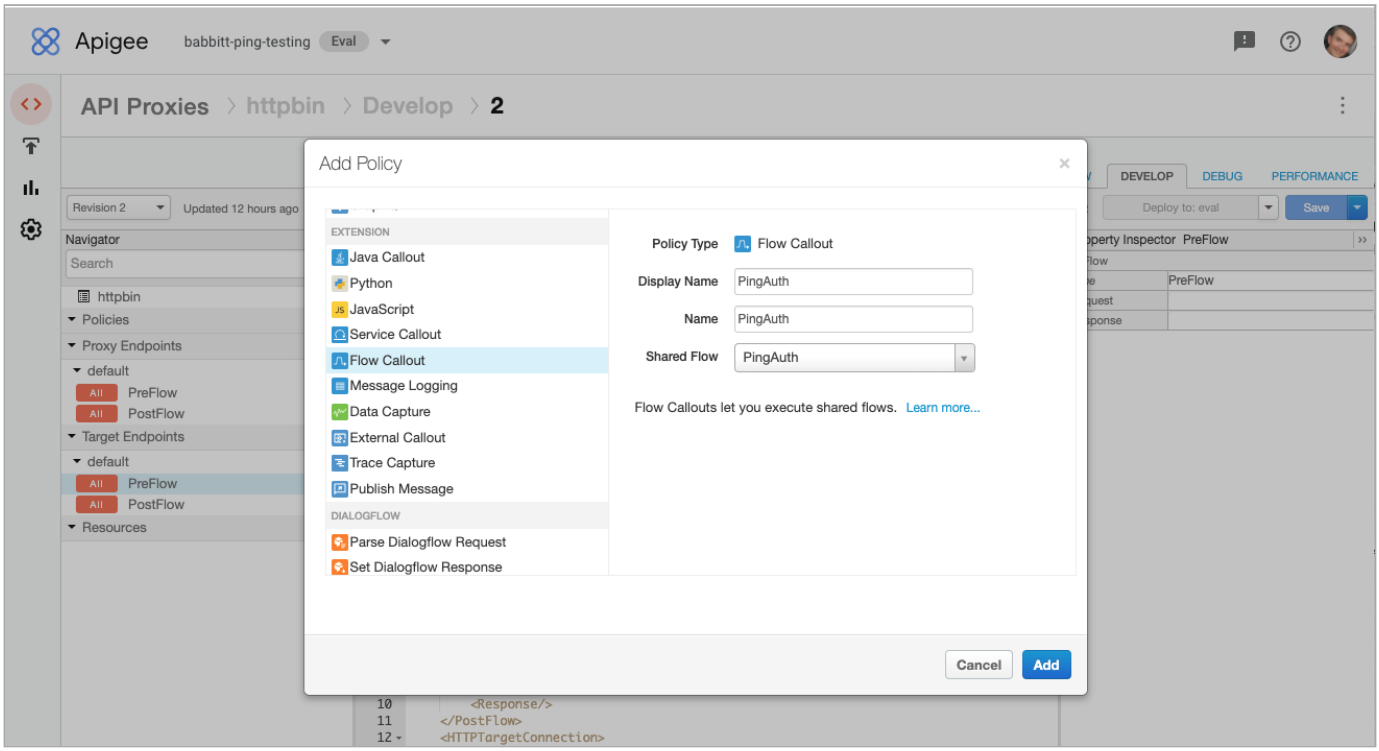
Configure the PingAuth shared flow on the API proxies where you want to use PingAccess as the external authorization policy runtime service.

Steps

1. Add a Flow Callout policy:
    1. In **Develop → API Proxies**, go to one of your APIs and click the **Develop** tab. Make sure that you are on the latest revision of the proxy.
    2. In the **Policies** section of the **Navigator**, click **+** to add a policy.

3. Add a **Flow Callout Policy**, and in the **Shared Flow** list, select **PingAuth**.

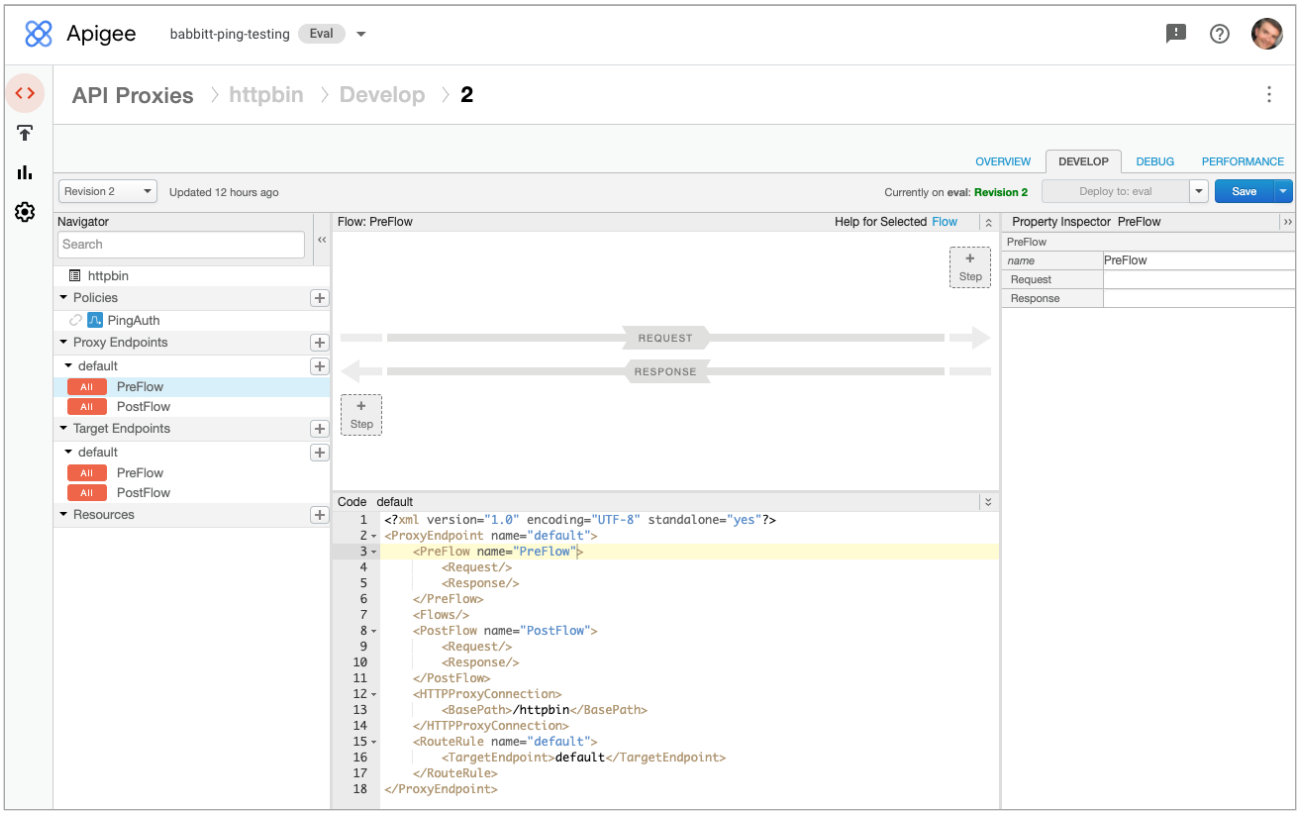
4. Click **Save**.



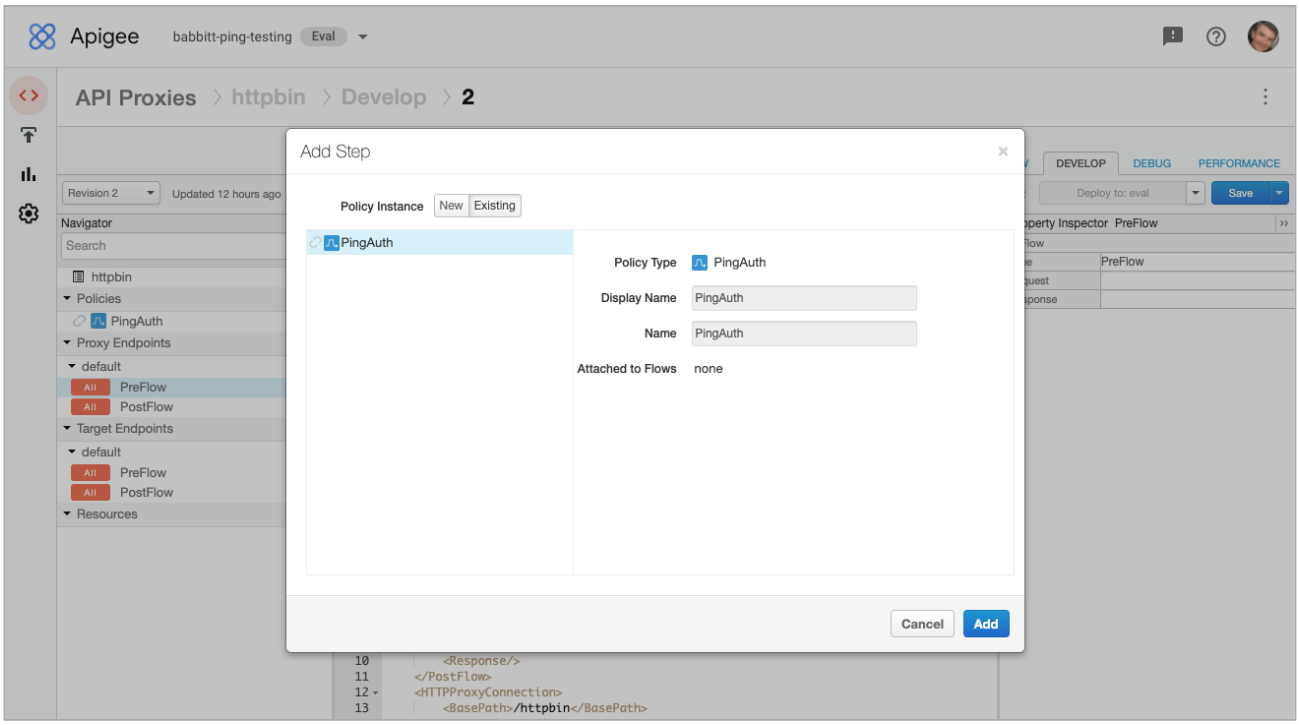
2. Attach the **Flow Callout Policy** to **Flows**.

> **Note**
>
> When PingAccess is integrated as the external authorization policy runtime service for Apigee, it should be integrated in the preflow of the request to the proxy endpoint, because the authentication and authorization function provided by PingAccess should occur before most other policies execute.
> You can consider other ways to integrate PingAccess by reading about flows at https://cloud.google.com/apigee/docs/api-platform/fundamentals/what-are-flows.

1. In the **Proxy Endpoint** section of the **Navigator**, click **PreFlow**, then click **+Step** to add a flow step to the request.

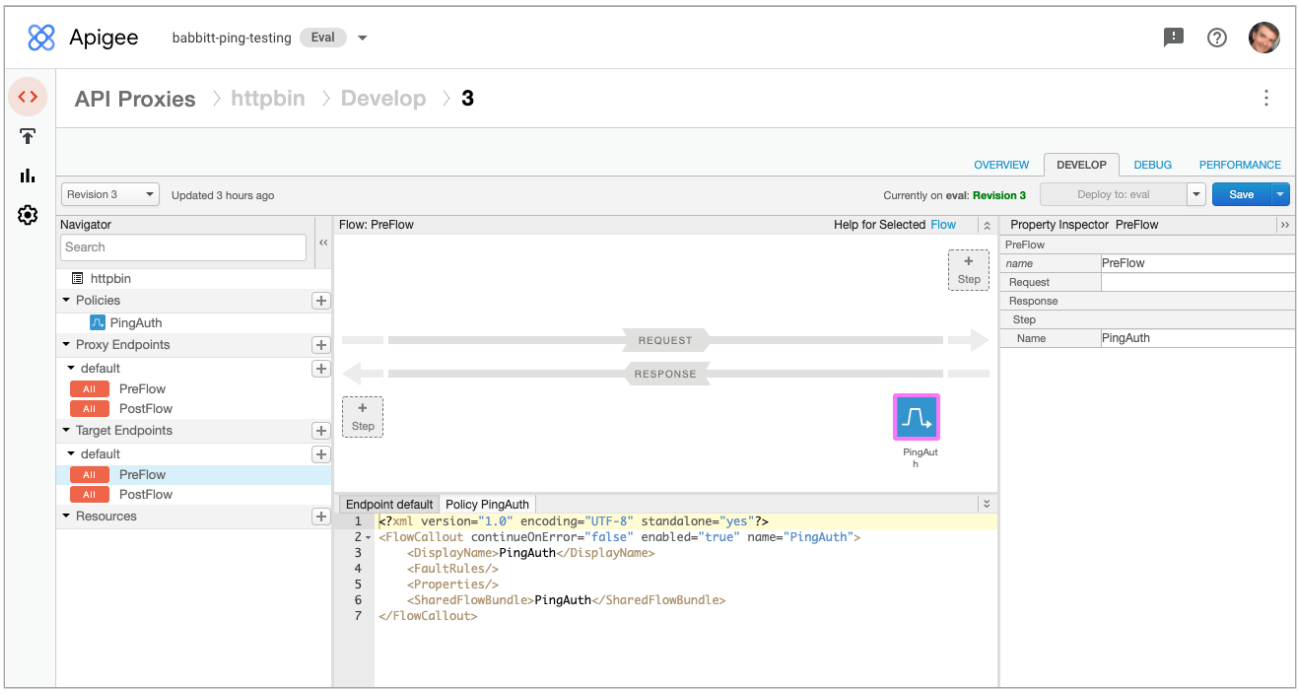


2. On the **Existing** tab, select the flow callout policy that you created, then click **Add**.



3. In the **Target Endpoint** section of the **Navigator**, select **PreFlow**, then add the flow callout policy as a **Step** to the **Response** flow.

This gives PingAccess an early opportunity to process the API response from the target API before it is processed by Apigee.



3. Save and deploy the updated proxy.

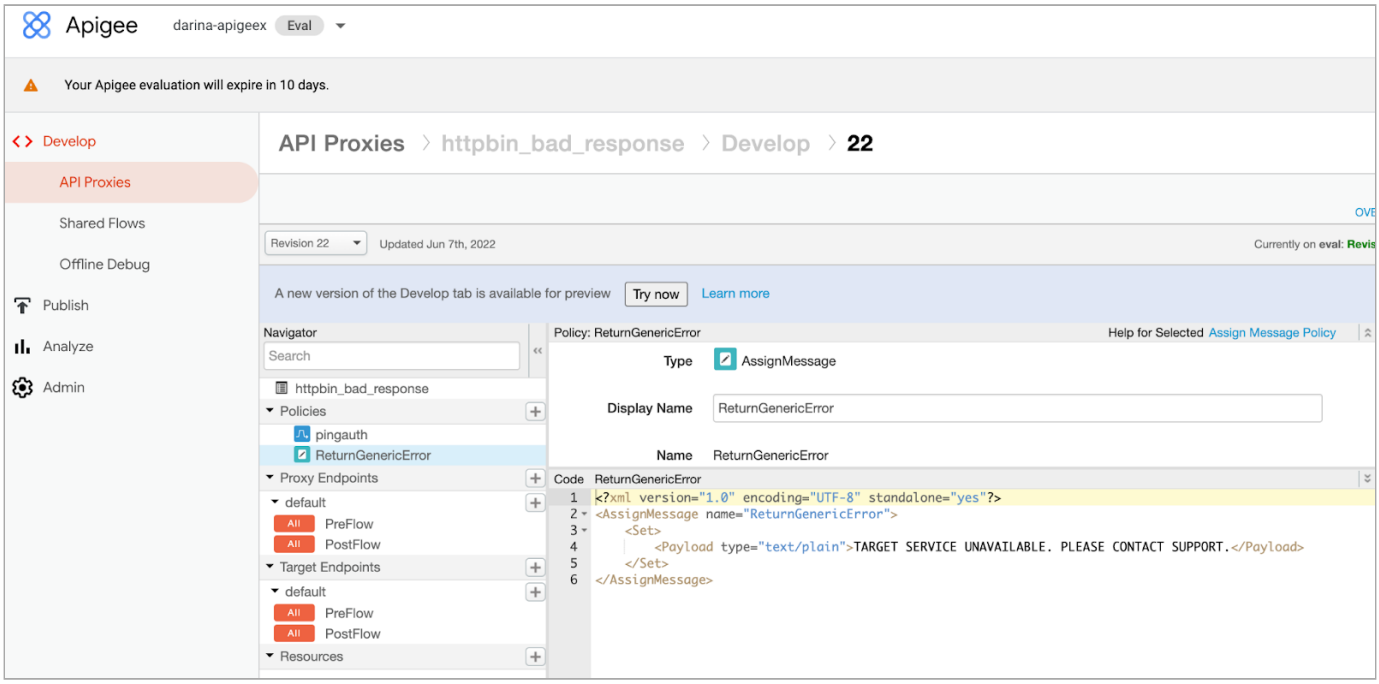
Creating an error handling policy in Apigee

About this task

When the target server returns an error, having an Apigee policy to handle that error prevents the API proxy from going into an error state in the target endpoint response and preventing the normal API proxy flow from continuing to the proxy endpoint.

Steps

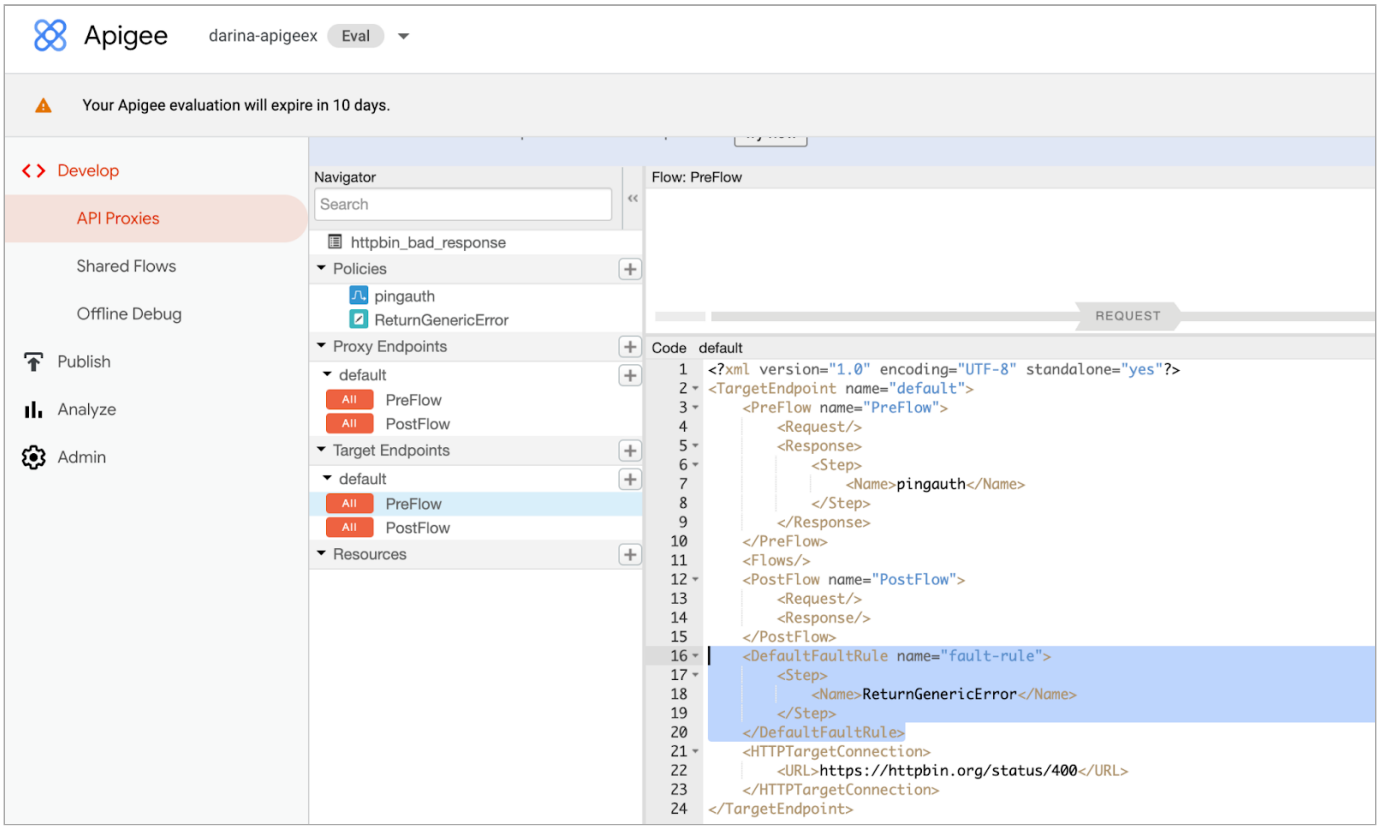
1. Add a policy with the **AssignMessage** type.



2. Edit the target endpoint and add the new policy as a step in the default fault rule.

   For example:

```
<DefaultFaultRule name="fault-rule">
     <Step>
          <Name>ReturnGenericError</Name>
     </Step>
</DefaultFaultRule>
```



Configuring API applications in PingAccess

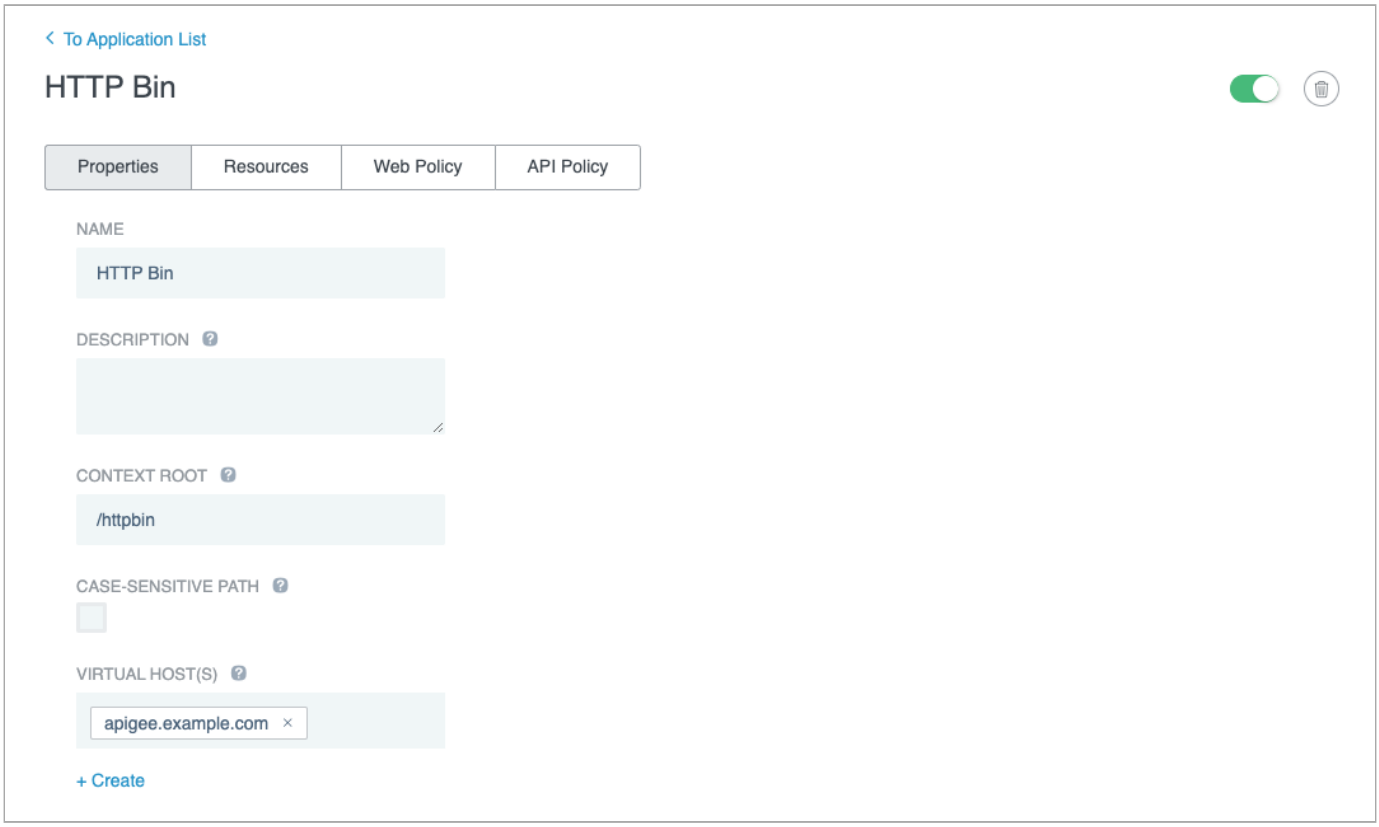
For each of the API proxies for which you attached the shared flow, configure a corresponding application in PingAccess.

About this task

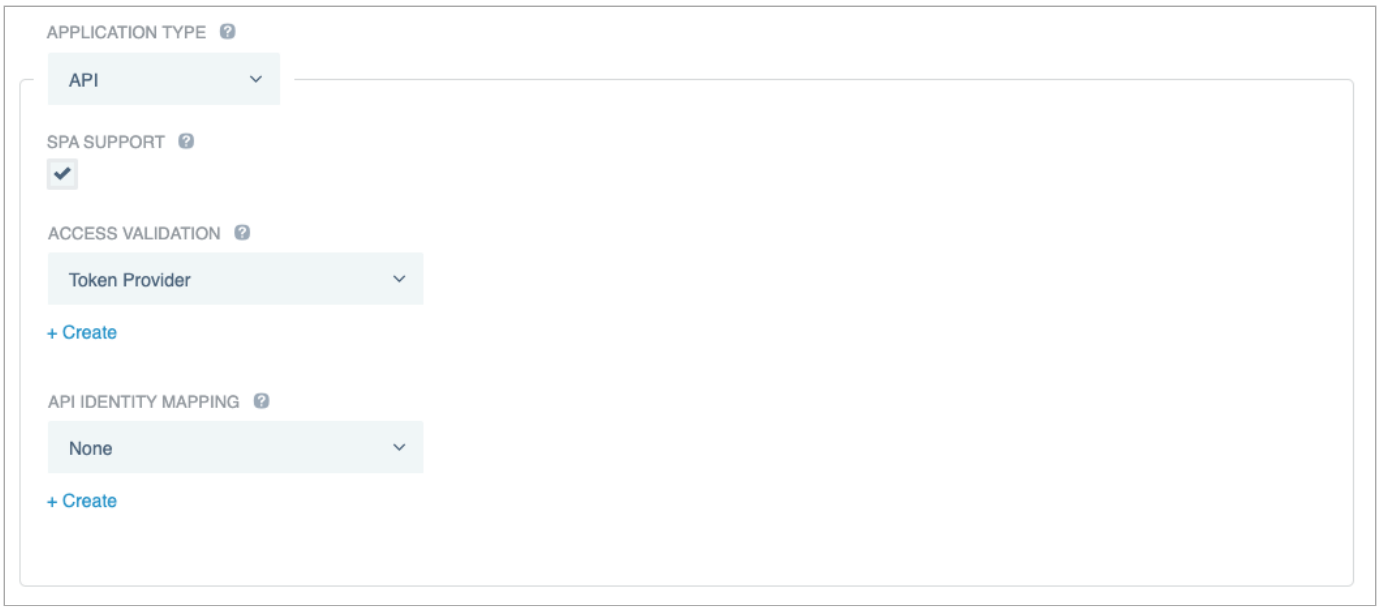
The application is where you manage access control rules, identity mappings, and other application protection features. To add an API application, configure access validation, and configure Apigee as the application destination:

Steps

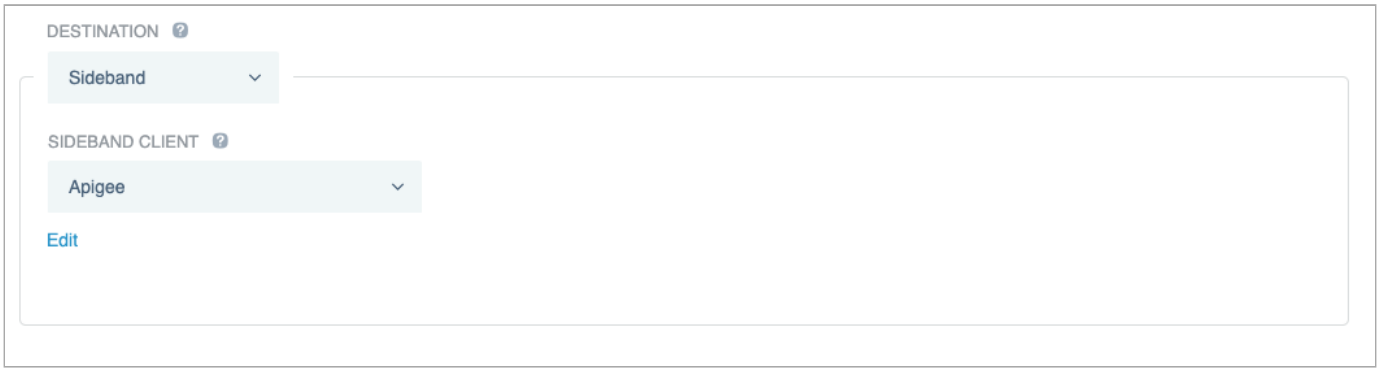
1. Go to **Applications → Applications** and click **+Application**.
2. Enter a **Name**, and then enter the **Context Root** and select or create the **Virtual Host(s)** values to match how the application's APIs are exposed from your Apigee environment.
    1. **Optional:** Click **+Create** to create a new **Virtual Host**.



3. In the **Application Type** list, select **API**.
4. In the **Access Validation list**, select a form of access validation.



5. To configure Apigee as the application destination, in the **Destination** section, select **Sideband**, and then select the **Sideband Client** that you created earlier.
6. Click **Save**.



Supporting Web+API Applications

PingAccess simplifies adding OpenID Connect (OIDC) and OAuth to API-based web applications, such as single-page applications (SPAs).

About this task

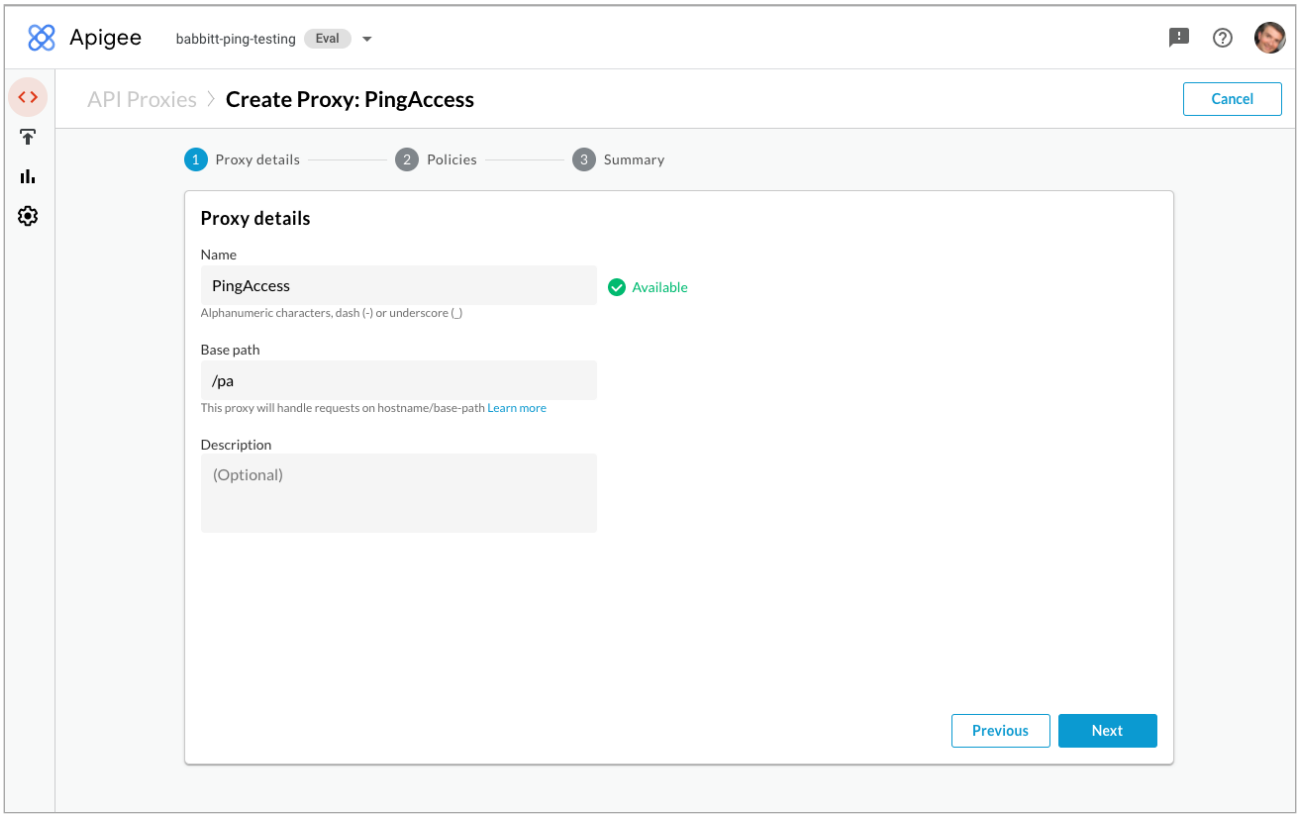
In this configuration, PingAccess completely manages the OIDC authentication for the SPA, maintains a cookie-based web session with the browser, and replaces the cookie for an OAuth access token (or other identity mappings) before invoking the target API. You must perform additional steps to support this configuration.

Steps

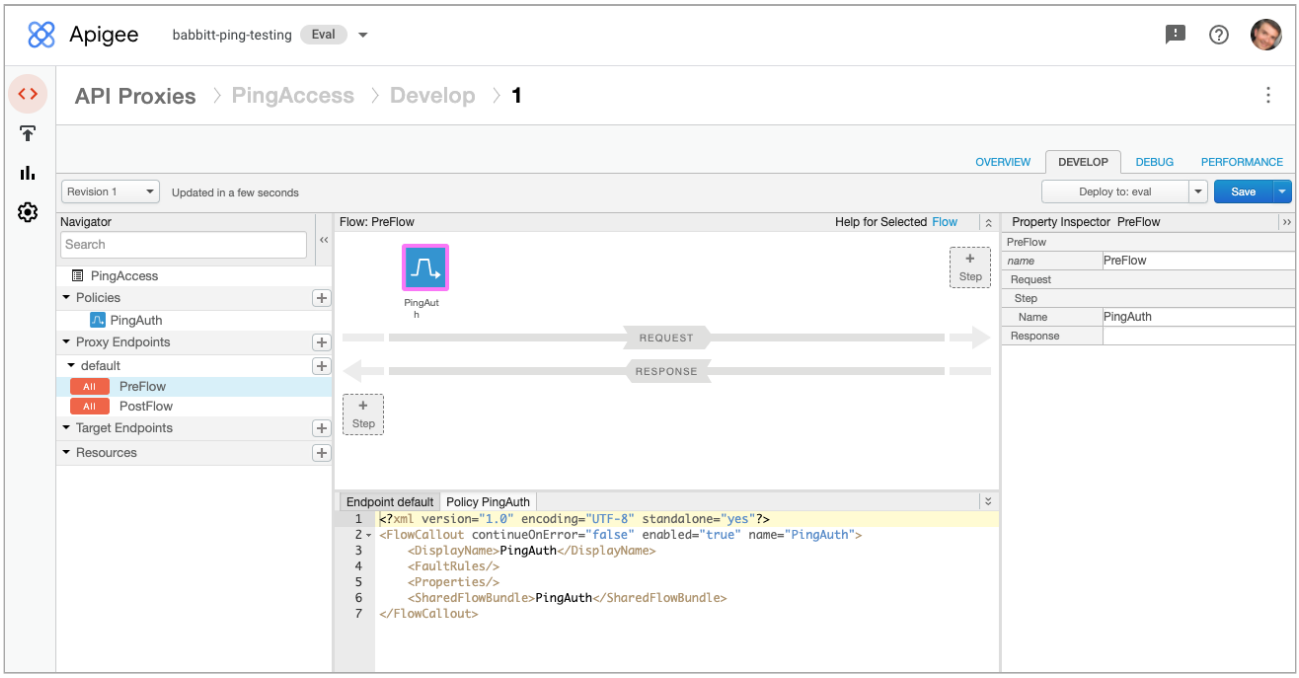
1. Configure Apigee to intercept calls for PingAccess.

> **Note**
>
> If you selected the **Use context root as reserved resource base path** check box on the PingAccess application you plan to use in conjunction with Apigee, skip ahead to step 2. When enabled, this feature provides reserved PingAccess resources from that application's context root, which makes step 1 unnecessary.

    1. In Apigee, go to **Develop → API Proxies** and click **Create New**.
    2. On the **Create Proxy** page, click **No Target**.
    3. In the **Name** field, enter `PingAccess`.
    4. In the **Base Path** field, enter `/pa`.



5. In the **Policies** section of the **Navigator**, click **+** to add a policy.
6. Add a **Flow Callout Policy**, and in the **Shared Flow** list, select **PingAuth**.
7. Click **Save**.
8. In the **Proxy Endpoints** section of the navigator, select **PreFlow**, then add the flow callout policy as a **Request Step**.



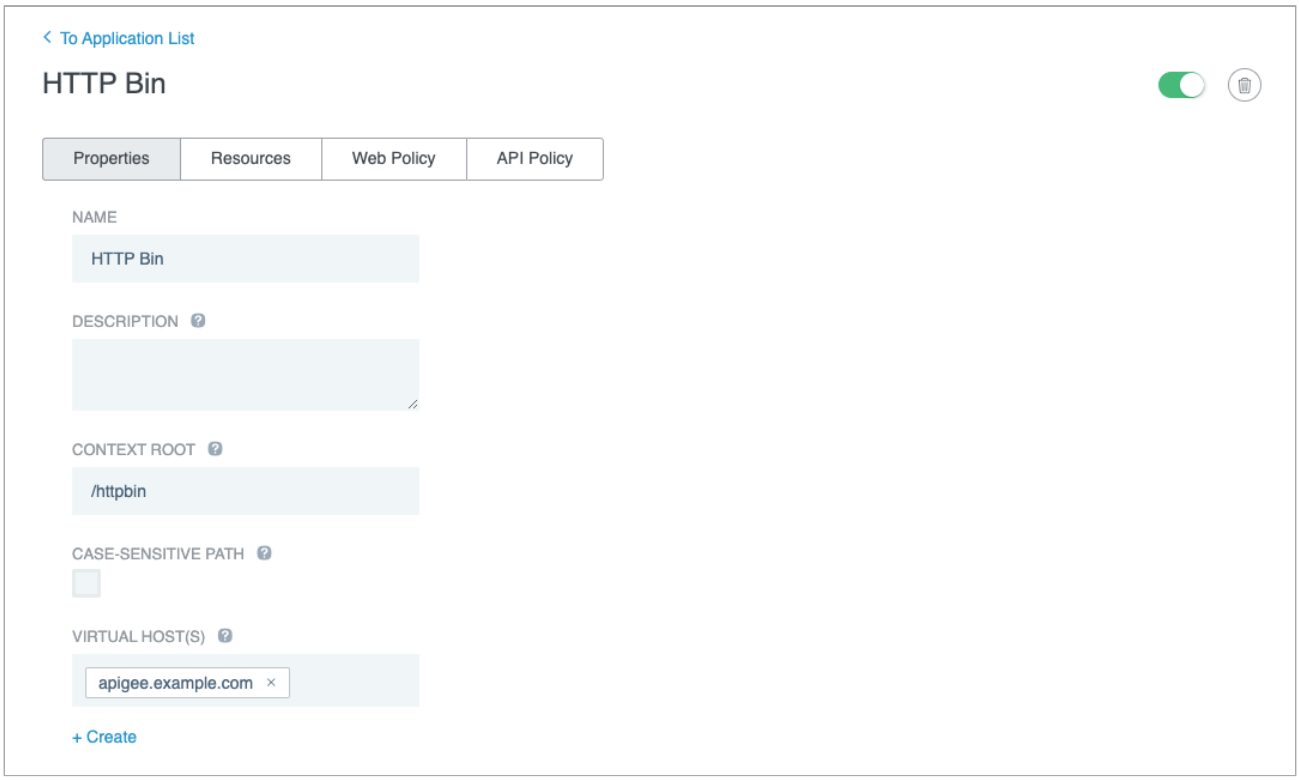
9. Save and deploy the new proxy.

2. Add a Web+API application in PingAccess:

    1. Go to **Applications → Applications** and click **+Application**.

    2. Enter a **Name**, and then enter the **Context Root** and select or create **Virtual Host(s)** values to match how the application's APIs are exposed from your Apigee environment.

    > **Note**
    >
    > To create a Virtual Host, click **+Create** below the field name.

3. Configure the web session:

    1. In the **Application Type** list, select **Web+API**.

    2. Under **Web Session**, click **+Create**.

    3. Enter the web session details, including the OIDC sign-on details configured in your OpenID Provider (OP).

    > **Note**
    >
    > PingAccess can only manage the OIDC authentication on behalf of the browser if PingAccess, through Apigee, is configured as the redirect URL in your OIDC provider.
    > For example, `https://apigee.example.com/pa/oidc/cb` .

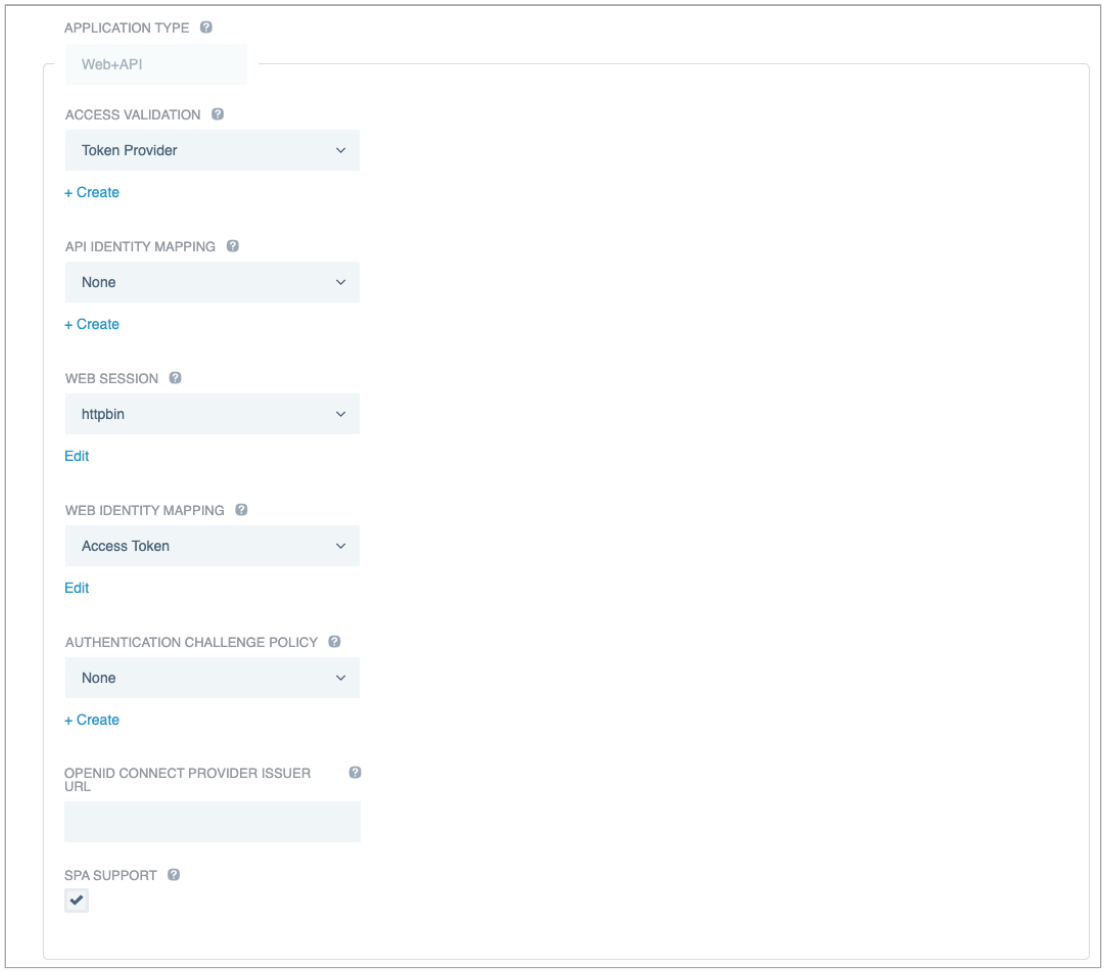
    4. Click **Save** to save the web session.

    5. Under **Web Identity Mapping**, click **+Create**.

6. Name the identity mapping Access Token and select the type **Web Session Access Token**.

   This configures PingAccess to forward the OAuth Access Token it obtains from the OIDC provider Authorization Server as the bearer token to the API behind Apigee.

7. Click **Save**.



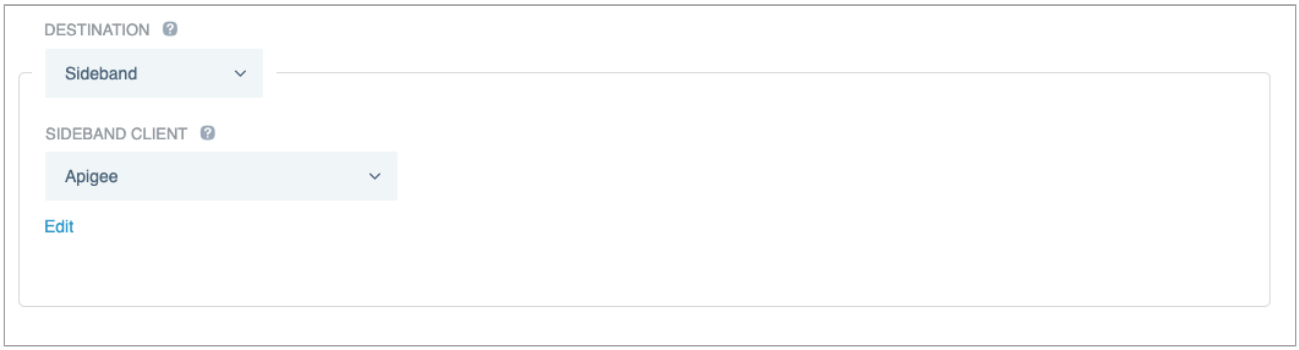
4. In the **Access Validation** list, select the form of access validation that will be applied for non-web API clients, such as mobile applications.

5. Configure Apigee as the application destination:

   1. In the **Destination** list, select **Sideband**.

   2. In the **Sideband Client** list, select the sideband client that you created earlier.

   3. Click **Save**.



PingOne Protect integration

Configure PingOne Protect risk evaluations in web applications protected by PingAccess and set PingAccess's response to the different levels of risk scores that might be returned.

PingOne Protect monitors end user requests and generates a risk score of low, medium, or high, based on the user's activity and device context.

You can create risk policies through the PingAccess administrative console or administrative application programming interface (API) to respond to PingOne Protect's evaluations dynamically. Risk policies tell PingAccess when to gather user device profile information and start a new evaluation. They also enable you to apply different access control or authentication challenge policies based on the risk score that PingOne Protect determines.

For example, if the determined risk score is `MEDIUM`, you might direct the user to complete step-up authentication with additional multi-factor authentication (MFA) methods. However, if the determined risk score is `HIGH`, you might completely deny access to the resource instead.

The following topics demonstrate how to integrate PingOne Protect evaluations within PingAccess and create PingAccess risk policies to leverage different responses to end user requests based on their determined risk score:

1. To configure a connection between PingOne Protect and PingAccess, see Adding a PingOne connection.
2. To configure a risk policy, see Adding a risk policy.
3. To assign a risk policy to a specific application or application resource, see Application field descriptions and Adding application resources.

PingAccess evaluates the risk policies assigned to each application and resource before processing application and resource-specific rules and rule sets.

1. If you enabled device profiling in your risk policy configuration, either PingAccess collects the user's device profile or the front end application collects the user's device profile and submits it to PingAccess.
2. PingAccess collects user data, such as user ID and user-agent IP address, then sends a risk evaluation request to PingOne Protect each time the *<riskCheckInterval>* that you specified in the risk policy passes.

> **Note**
>
> By default, PingAccess doesn't collect user behavioral data when the **Device Profiling Method** is **Captured by PingAccess**.

3. After PingOne Protect returns an evaluation, PingAccess follows the response specified in the risk policy for the determined risk score or for a failed evaluation if PingAccess is unable to gather the expected risk response from PingOne Protect.

   For example, if the returned risk score is `HIGH` and the risk policy specifies that an authentication challenge policy (ACP) should be issued to an end user with a high-risk score, then PingAccess returns the authentication challenge defined by the ACP to the end user for reauthentication.

   If the risk policy specifies that PingAccess should follow a specific rule or rule set, then it uses that rule or rule set to determine whether the end user should be allowed access.

4. If the end user meets the constraints set by the risk policy, PingAccess moves on to the next stage of request processing, reviewing the normal policies attached to the application or resource.

   If a request is denied, PingAccess uses the rejection handler set within the risk policy or follows the rejection response specified in the rules or rule sets attached to the application or resource. The former case is applicable if the risk policy was set to deny an end user's request. Otherwise, the rejection response specified in the normal policy takes precedence.

5. After PingAccess has completed request processing, it sends PingOne Protect feedback on whether an end user's request was allowed or denied.

Token Providers

This section provides instructions on how to set up functionality between PingAccess and several common token providers.

- If you're using PingFederate, see Configure PingFederate as the token provider for PingAccess.
- If you're using the QuickStart Utility in conjunction with PingFederate and PingAccess, see Use the PingAccess QuickStart utility.
- If you're using PingOne, see Protect applications using PingAccess and PingOne for Customers.
- If you're using Azure AD, see PingAccess for Azure AD.

Configure PingFederate as the token provider for PingAccess

This section explains how to manually configure PingAccess and PingFederate to work together, with PingAccess as the access manager and PingFederate as the token provider.

For more information, see the following topics:

- Configure PingFederate for PingAccess connectivity
- Connect PingAccess to PingFederate

The features documented here are affected by the settings in the configuration file. See the Configuration file reference for more information.

Configure PingFederate for PingAccess connectivity

This section explains how to configure PingFederate for PingAccess connectivity.

This configuration procedure covers the following:

1. Enabling PingFederate roles and protocols
2. Creating a password credential validator
3. Configuring an IdP adapter
4. Defining the default scope
5. Creating an access token manager
6. Configuring an IdP adapter mapping
7. Configuring an access token mapping
8. Creating an OpenID Connect policy
9. Creating a resource server client
10. Creating a web session client
11. Creating and exporting a certificate

> **Important**
>
> These steps assume you have installed PingFederate 10.1. If you are using an earlier version of PingFederate, the steps might differ. The example assumes that your PingFederate instance is available at https://*<mypingfedserver>*, using ports `9031` and `9999` respectively for the runtime and administration functions.
> These steps assume you have installed PingAccess 6.1. If you are using an earlier version of PingAccess, the steps might differ. This example assumes that your PingAccess instance is available at https://*<mypingaccessserver>* and that `3000` is the default listening port.

Enabling PingFederate roles and protocols

If you are using PingFederate 10.0 or earlier, ensure that PingFederate is configured to respond to OAuth and OpenID Connect (OIDC)requests.

About this task

For more information on PingFederate roles and protocols, see Choose roles and protocols.

Steps

1. In the PingFederate administrative console, go to **System → Server → Protocol Settings**.
2. Click **Roles & Protocols** and verify that the following items are selected. Click **Next**.
    - **Enable OAuth 2.0 Authorization Server as Role** (role) and **OpenID Connect** (protocol)
    - **Enable Identity Provider (IdP) Role and Support the Following:** (role) and **SAML 2.0** (protocol)
3. On the **Federation Info** tab, enter the URL of your PingFederate environment and your Security Assertion Markup Language (SAML) 2.0 entity ID, then click **Next**.

    For example:

    - **Base URL**: `https://mypingfedserver:9031`
    - **SAML 2.0 Entity ID**: `https://mypingfedserver/idp`
4. Review the summary. Click **Save**.

Next steps

Create a password credential validator.

Creating a password credential validator

Create a password credential validator (PCV) and then create a username and password to use in authentication.

About this task

For more information on PCVs, see Configuring the Simple Username Password Credential Validator.

Steps

1. Go to **System → Data & Credential Stores → Password Credential Validators**.

2. Click **Create New Instance**.
3. In the **Instance Name** field, enter an instance name of your choosing.

   For example, `My_PCV`.
4. In the **Instance ID** field, enter an instance ID of your choosing.

   For example, `mypcv`.
5. From the **Type** list, select **Simple Username Password Credential Validator**, and then click **Next**.
6. On the **Instance Configuration** tab, click **Add a new row to 'Users'**.
7. In the **Username** field, enter a username.
8. In the **Password** fields, enter and confirm a password.
9. Click **Update**, then click **Next**.
10. On the **Summary** tab, click **Save**.

Next steps

Configure an IdP adapter.

Configuring an IdP adapter

Configure an identity provider (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 (PCV).

About this task

For more information, see Configuring an IdP adapter instance.

Steps

1. Go to **Authentication → Integration → IdP Adapters**.
2. Click **Create New Instance**.
3. In the **Instance Name** field, enter an instance name of your choosing.

   For example, `My_IdP`.
4. In the **Instance ID** field, enter an instance ID of your choosing.

   For example, `myidp`.
5. From the **Type** list, select **HTML Form IdP Adapter**, and then click **Next**.
6. On the **IdP Adapter** tab, under **Password Credential Validator Instance**, click **Add a new row to 'Credential Validators'.**
7. From the **Password Credential Validator Instance** list, select the password credential validator you created previously, for example, **My_PCV**, and then click **Update**.
8. Click **Next** until the **Adapter Attributes** tab is displayed.

9. Locate the `username` attribute, then select the **Pseudonym** check box.
10. Click **Next** until the **Summary** tab is displayed. Click **Save**.

Next steps

Define the default scope.

Defining the default scope

Use the **Scope Management** section to define the default scope.

About this task

For more information, see Define scopes.

Steps

1. Go to **System → OAuth Settings → Scope Management**.
2. Click the **Common Scopes** tab, then 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. Click **Next** until you reach the **Default Scope** tab.
4. On the **Default Scope** tab, enter a description.

   For example, `default scope`.
5. Click **Save**.

Next steps

Create an access token manager.

Creating an access token manager

Create an access token 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.

About this task

For more information, see OAuth Access token management.

Steps

1. Go to **Applications → OAuth → Access Token Management**.
2. Click **Create New Instance**.
3. In the **Instance Name** field, enter an instance name of your choosing.

   For example, `General Access Token`.
4. In the **Instance ID** field, enter an instance ID of your choosing.

   For example, `GeneralAccessToken`.
5. From the **Type** list, select **Internally Managed Reference Tokens**.
6. Click **Next** until the **Access Token Attribute Contract** tab is displayed.
7. In the **Extend the Contract** field, enter `UserName`, and then click **Add**.
8. Click **Next** until the **Summary** tab is displayed. Click **Save**.

Next steps

Configure an IdP adapter mapping.

Configuring an IdP adapter mapping

Configure an identity provider (IdP) adapter mapping to map attributes.

About this task

For more information, see Managing IdP adapter grant mapping.

Steps

1. Go to **Authentication → OAuth → IdP Adapter Grant Mapping**.
2. From the **Source Adapter Instance** list, select the adapter you created in Configuring an IdP adapter.
3. Click **Add Mapping**, then click **Next** until the **Contract Fulfillment** tab is displayed.
4. In the **USER_KEY** row:
    1. In the **Source** column, select **Adapter**.
    2. In the **Value** column, select **username**.
5. In the **USER_NAME** row:
    1. In the **Source** column, select **Adapter**.
    2. In the **Value** column, select **username**.

6. Click **Next** until the **Summary** tab is displayed. Click **Save**.

Next steps

Configure an access token mapping.

Configuring an access token mapping

Configure an access token mapping that maps attributes to be requested from the OAuth resource server with the corresponding access token.

About this task

For more information, see Managing access token mappings.

Steps

1. Go to **Applications → OAuth → Access Token Mapping**.
2. From the **Context** list, select **Default** or select your identity provider (IdP) adapter instance.
3. From the **Access Token Manager** list, select the access token you created in Creating an access token manager.

   For example, **GeneralAccessToken**.
4. Click **Add Mapping**. Click **Next**.
5. On the **Contract Fulfillment** tab, from the **Source** list, select **Persistent Grant**.
6. From the **Value** list, select **USER_KEY**.
7. Click **Next** until the **Summary** tab is displayed. Click **Save**.

Next steps

Create an OpenID Connect policy.

Creating an OpenID Connect policy

Configure an OpenID Connect (OIDC) policy to define OIDC policies for client access to attributes mapped according to OpenID specifications.

About this task

For more information, see Configuring OpenID Connect policies.

Steps

1. Go to **Applications → OAuth → OpenID Connect Policy Management**.
2. Click **Add Policy**.
3. In the **Policy ID** field, enter an Policy ID of your choosing.

   For example, `OIDC`.

4. In the **Name** field, enter a name of your choosing.

   For example, `OIDC`.

5. From the **Access Token Manager** list, select the access token you created in Configuring an access token mapping.

   For example, **GeneralAccessToken**.

6. Click **Next**.
7. On the **Attribute Contract** tab, delete all items beneath the **Extend the Contract** heading.
8. Click **Next** until the **Contract Fulfillment** tab is displayed.
9. From the **Source** list, select **Access Token**.
10. From the **Value** list, select **username**.
11. Click **Next** until the **Summary** tab is displayed. Click **Save**.
12. In the **Action** column for the policy you created, if the policy is not already listed as the default, click **Set as Default**.

Next steps

Create a resource server client.

Creating a resource server client

Configure an OAuth client for use with PingFederate token provider resource server configuration in PingAccess.

About this task

For more information, see Manage OAuth clients.

Steps

1. Go to **Applications → OAuth → Clients**.
2. Click **Add Client**.
3. In the **Client ID** field, specify a client ID.

   ```
   pa_rs
   ```

4. In the **Name** field, specify a name.

   ```
   PingAccessResourceServer
   ```

5. In the **Client Authentication** section, select **Client Secret**.
6. In the **Client Secret** section, select **Change Secret**, and then click **Generate Secret**.

> **Tip**
> Copy the secret to a secure location so that you can use it in PingAccess configuration.

7. In the **Redirect URIs** field, enter the OpenID Connect (OIDC) callback redirect to the PingAccess server.

   For example, `https://mypingaccessserver:3000/pa/oidc/cb`.

8. Click **Add**.
9. In the **Allowed Grant Types** section, select the **Access Token Validation (Client is a Resource Server)** check box.
10. Click **Save**.

Next steps

Create a web session client.

Creating a web session client

Configure an OAuth client for use with web session configuration in PingAccess.

About this task

For more information, see Manage OAuth clients.

Steps

1. Go to **Applications → OAuth → Clients**.
2. Click **Add Client**.
3. In the **Client ID** field, specify a client ID.

   ```
   pa_wam
   ```

4. In the **Name** field, specify a name.

   ```
   PingAccessWebAccessManagement
   ```

5. In the **Client Authentication** section, select **Client Secret**.
6. In the **Client Secret** section, select **Change Secret**, and then click **Generate Secret**.

> **Tip**
> Copy the secret to a secure location so that you can use it in PingAccess configuration.

7. In the **Redirect URIs** field, add the OpenID Connect (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. In the **Allowed Grant Types** section, select the **Authorization Code** check box.
11. Click **Save**.

Next steps

Create and export a certificate.

Creating and exporting a certificate

Create and export a certificate for the PingFederate server that you will import to PingAccess to establish trust.

About this task

For more information, see Manage SSL server certificates.

Steps

1. In the PingFederate administrative console, go to **Security → Certificate & Key Management → SSL Server Certificates**.
2. Click **Create New**.
3. In the **Common Name** field, enter the PingFederate server address.

   For example, `mypingfedserver`.
4. In the **Organization** field, enter your organization's name.
5. In the **Country** field, enter the two-letter abbreviation for your country.
6. Complete the remaining fields as required.
7. Click **Next**.
8. Click **Save**.
9. In the **Action** section, click **Activate Default for Runtime Server**.
10. In the **Action** section, click **Export**.
11. Select **Certificate Only**. Click **Next**.
12. Click **Export**, and then save the exported certificate.
13. Click **Done**.

Next steps

Connect PingAccess to PingFederate and configure an application.

Connect PingAccess to PingFederate

This section explains how to configure PingAccess to communicate with PingFederate.

In this configuration procedure, you will perform the following tasks:

1. Import certificates and create a trusted certificate group.
2. Configure the token provider.

After configuring PingAccess to use PingFederate as a token provider, you can configure it to protect a web application. See Protecting a web application for more information.

Importing certificates and creating a trusted certificate group

Import a certificate for the PingFederate server to establish trust.

About this task

For more information, see Certificates.

Steps

1. Click **Security**, then go to **Certificates > Certificates**.
2. Click **+ Add Certificate**.
3. In the **Alias** field, enter an alias for the certificate.

   For example, `PingFed`.
4. To select the certificate, click **Choose File**.
5. To import the certificate, click **Add**.

   Result:

   A new certificate row appears on the **Certificates** tab.
6. Click **Security**, then go to **Certificates > Trusted Certificate Groups**.
7. Click **+ Add Trusted Certificate Group**.
8. Drag a certificate onto the box that appears.
9. In the **Name** field, enter a name for the group in the box that appears.

   For example, `PingFed`.
10. Click **Save**.

Next steps

Configure the token provider.

Configuring the token provider

Establish communication with the token provider, PingFederate.

About this task

For more information, see Manage Token Provider.

Steps

1. Click **Settings**, then go to **System > Token Provider > PingFederate > Runtime**.
2. In the **Issuer** field, enter the PingFederate issuer URI.
3. From the **Trusted Certificate Group** list, select the **PingFed** certificate group.
4. Click **Save**.
5. Click **Settings**, then go to **System > Token Provider > PingFederate > Administration**.
6. In the **Host** field, enter the host name or Internet Protocol (IP) address for the PingFederate Admin.

   For example, `mypingfedserver`.
7. In the **Port** field, enter the port number for the PingFederate Admin.

   For example, `9999`.
8. In the **Admin Username** field, enter the username.

   This username only requires auditor (read only) permissions in PingFederate.
9. In the **Admin Password** field, enter the password.
10. From the **Secure** list, select **Secure**.
11. From the **Trusted Certificate Group** list, select the **PingFed** certificate group.
12. Click **Save**.
13. Click **Settings**, then go to **System > Token Provider > PingFederate > OAuth Resource Server**.
14. In the **Client ID** field, enter the OAuth Client ID you defined when creating the PingAccess OAuth client in PingFederate.

    For example, `pa_rs`.
15. In the **Client Credentials Type** section, select **Secret**, then enter the **Client Secret** assigned when you created the PingAccess OAuth client in PingFederate.
16. 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`.
17. Click **Save**.

Next steps

You can configure PingAccess to Protect a web application.

Use the PingAccess QuickStart utility

This section explains how to protect a web-based application using PingAccess as the access manager, PingFederate as the token provider, and the PingAccess QuickStart utility to enable the connection and provide sample applications.

The QuickStart utility is designed to support a sample environment and aid in your understanding of how PingAccess and PingFederate work together to protect applications. You can perform the example configuration to achieve a working result and become familiar with this solution, or you can substitute your own data.

The QuickStart utility creates 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

The QuickStart utility does not retain information about the PingAccess and PingFederate configuration. If you restart the utility, you must repeat the configuration steps before using the sample applications.

This document does not detail the usage of identity mappings or authentication requirements that are common components of a typical configuration. Because 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.

This document does not discuss how to manually configure PingAccess and PingFederate to work together, since the QuickStart application automates this process. See Configure PingFederate as the token provider for PingAccess for the manual steps.

The following topics are covered in this section:

- Installing and configuring QuickStart components
- Connecting the QuickStart utility to PingAccess and PingFederate
- Using sample applications
- Restoring PingFederate or PingAccess

Installing and configuring QuickStart components

Install the QuickStart utility along with PingAccess and PingFederate.

Steps

1. Download PingFederate 10.3 or later.
2. Install PingFederate as shown in Installing and uninstalling PingFederate.
3. **Optional:** If you plan to use the one-time authentication app, install the client-initiated backchannel authentication (CIBA) authenticator plugin.
    1. Copy the contents of the `<Quickstart Home>/plugins/deploy` directory into the `<PingFederate Home>/server/default/deploy` directory.

2. Copy the contents of the `<Quickstart Home>/plugins/conf` directory into the `<PingFederate Home>/server/default/conf` directory.

> **Note**
>
> If you want to use an authenticator other than the CIBA authenticator, you must manually configure it in PingFederate before configuring the one-time authentication app.

4. Perform the first-time configuration
5. Download PingAccess 6.0 or later.
6. Install PingAccess and perform the first-time configuration.
7. Download and extract the PingAccess QuickStart bundle.
8. Change to the QuickStart directory and run the `quickstart-server-<version>.jar` file.

```
java -jar quickstart-server-6.0.0.0.jar
```

You can use the `--server.port=<port>` argument to specify a port other than the default of 8443.

```
java -jar quickstart-server-6.0.0.0.jar --server.port=8444
```

Next steps

Connecting the QuickStart utility to PingAccess and PingFederate

Connecting the QuickStart utility to PingAccess and PingFederate

Connect the QuickStart utility to your installed PingAccess and PingFederate deployments.

Steps

1. Go to https://*hostname*:8443 and sign on to the QuickStart utility.

> **Note**
>
> If you cannot access the utility, you might need to restart it by rerunning the `.jar` file. For more information, see the final step in Installing and configuring QuickStart components.

Result:

The QuickStart user interface is displayed.

2. Click **Connect**.
3. Enter the PingFederate runtime configuration.
    1. In the **Host** field, enter the host name.
    2. In the **Port** field, enter the port.

4. Enter the PingFederate admin configuration.
    1. In the **Host** field, enter the host name.
    2. In the **Port** field, enter the port.
5. In the **Username** and **Password** fields enter the admin credentials for PingFederate, and then click **Validate**.
6. Click **Next**.
7. Enter the PingAccess runtime configuration.
    1. In the **Host** field, enter the host name.
    2. In the **Port** field, enter the port.
8. Enter the PingAccess admin configuration.
    1. In the **Host** field, enter the host name.
    2. In the **Port** field, enter the port.
9. In the **Username** and **Password** fields, enter the admin credentials for PingAccess, and then click **Validate**.
10. Click **Save and Close**.

Next steps

Use sample applications.

Using sample applications

Configure and launch one or more sample applications.

About this task

Once an application is configured, users can sign on using any configured user credentials. The QuickStart utility uses five users, which are configured by default. Credentials for these users are displayed in the **User Credentials** section.

Steps

1. Go to https://<*hostname*>:8443 and sign on to the QuickStart utility.

    > **Note**
    >
    > If you cannot access the utility, you might need to restart it by rerunning the `.jar` file. For more information, see the final step in Installing and configuring QuickStart components.

2. Click **Sample Apps**.
3. In the **Access Control** section, under one of the sample applications, click **Configure**.

    > **Note**
    >
    > If you have already configured another sample application, the utility skips the PingFederate and PingAccess configuration pages.

Result:

The **Configure PingFederate** window is displayed.

4. If you have not yet configured an application, configure PingFederate and PingAccess for the sample applications.

   The user interface displays the details of each PingFederate and PingAccess configuration step.

   1. Click **Configure PingFederate**.
   2. Click **Next**.
   3. Click **Configure PingAccess**.
   4. Click **Next**.

5. Click **Configure PingAccess**.
6. Click **Save and Close**.
7. Click **Launch**.

   Result:

   The application launches in a new tab. You can sign on to the app using any configured set of credentials.

Sample app reference

This list shows the characteristics of each sample app included in the PingAccess QuickStart utility.

Sample apps

Traditional App

A web application that renders its views on the server side in response to HTTP requests. Once accessed, it displays a simple to-do list.

Single Page App

An application that uses Javascript to render different views within the browser. Once accessed, it displays a simple to-do list.

API-Only App

An application that is intended to be accessed with APIs and not through a UI. It lets users create and manage a simple to-do list.

One-Time Auth App

An application that has a resource that requires authorization for every request. It lets users send hypothetical money to specified recipients.

Viewing apps without access control

View the sample apps without any access control to understand their behavior.

Steps

1. Go to https://*hostname*:8443 and sign on to the QuickStart utility.

   > **Note**
   >
   > If you cannot access the utility, you might need to restart it by rerunning the `.jar` file. For more information, see the final step in Installing and configuring QuickStart components.

2. Click **Sample Apps**.
3. In the **No Access Control** section, under one of the sample applications, click **Launch**.

   Result:

   The app is displayed.

Restoring PingFederate or PingAccess

Restore PingFederate or PingAccess from a saved configuration file.

Steps

1. Go to https://*hostname*:8443 and sign on to the QuickStart utility.

   > **Note**
   >
   > If you cannot access the utility, you might need to restart it by rerunning the `.jar` file. For more information, see the final step in Installing and configuring QuickStart components.

2. Click **Sample Apps**.
3. In the **Archive Restoration** section, click **Upload PingFederate Archive** or **Upload PingAccess Archive**.
4. Select the relevant archive.
5. Click **Restore PingFederate Instance** or **Restore PingAccess Instance**.

Result

The specified instance is restored.

Protect applications using PingAccess and PingOne for Customers

Configure PingAccess to provide secure external access to applications using PingAccess and PingOne.

In this scenario, PingAccess provides an external path to applications while PingOne 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 might require configuring:
  - A virtual host
  - A web session or access token validator
  - A site
  - An application

After you complete the configuration, you can test the application using the virtual host and context root that you assign to it in PingAccess.

Configuring PingAccess to use PingOne for Customers as the token provider

Configure PingAccess to use PingOne as the token provider in the PingAccess user interface.

Before you begin

- Install PingAccess and verify that you can access the administrative console. For more information on installing PingAccess, see Installing and Uninstalling PingAccess.

> **Note**
>
> The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:
>
> ```
> Username: Administrator
> Password: 2Access
> ```

- Configure an application in PingOne.

About this task

For more information on configuring PingOne as the token provider, see Configuring PingOne.

Steps

1. Click **Settings**, then go to **System > Token Provider > PingOne**.
2. In the **Issuer** field, enter the PingOne Issuer URL.

   To obtain the Issuer URL, in PingOne, go to the **Configuration** tab of an application and copy the **Issuer** value.
3. **Optional:** In the **Description** field, enter a description for the connection.
4. From the **Trusted Certificate Group** list, select a trusted certificate group that PingAccess will use when authenticating to PingOne.
5. To configure the connection to use a configured proxy, click **Show Advanced** and select **Use Proxy**.
6. Click **Save**.

Configuring a PingAccess application

Perform the following steps to configure PingAccess applications.

Before you begin

- Install PingAccess and verify that you can access the administrative console. For information on installing PingAccess, see Installing and Uninstalling PingAccess.

> **Note**
>
> The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:
>
> ```
> Username: Administrator
> Password: 2Access
> ```

- Configure an application in PingOne.
- Configure PingAccess to use PingOne as the token provider.

About this task

For each application that you want to configure:

Steps

1. Create a virtual host.

   For more information on creating a virtual host, see Creating new virtual hosts.

   1. Click **Applications**, then go to **Applications > Virtual Hosts**.
   2. Click **+ Add Virtual Host**.
   3. In the **Host** filed, enter a 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 can also be specified, such as `*.example.com` .
   4. In the **Port** field, enter the port number for the virtual host.

      For example: `1234` .
   5. In the **Agent Resource Cache TTL (s)** field, indicate the number of seconds the agent can cache resources for this application.

      > **Note**
      >
      > Only applies to a destination of type `Agent` .
   6. Click **Save**.
2. Create a web session.

For more information on creating a web session, see Creating web sessions.

> **Note**
>
> A web session is only used when protecting a web application. To protect APIs, configure an access token validator.

1. Click **Access**, then go to **Web Sessions > Web Sessions**.
2. Click **+ Add Web Session**.
3. In the **Name** field, enter a name for the web session.
4. From the **Cookie Type** list, select your cookie type, either **Signed JWT** or **Encrypted JWT**.
5. In the **Audience** field, enter a unique value.
6. In the **Client ID** field, enter the PingOne client ID.

   > **Tip**
   >
   > You can find the Client ID on the **Profile** tab of the application you created.

7. From the **Client Credentials Type** list, select **Secret**.
8. In the **Client Secret** field, enter the client secret found on the application's **Configuration** tab.
9. Click **Show Advanced**.
10. In the **Scopes** section, specify one or more scopes.

    > **Note**
    >
    > Ensure the scopes you specify match those configured for the PingOne application. Find the scopes on the **Access** tab of your PingOne application.

11. Click **Save**.

3. Create a site.

For more information on creating a site, see Adding sites.

> **Note**
>
> In some configurations, a site might contain more than one application. A site can be used with more than one application, where appropriate.

1. Click **Applications**, then go to **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. From the **Secure** list, select whether or not the target is expecting secure connections.
6. If the target is expecting secure connections, from the **Trusted Certificate Group** list, select **Trust Any**.
7. Click **Save**.

4. Create an application in PingAccess for each application that you want to protect.

   For more information on creating an application, see Adding an application.

   1. Click **Applications**, then go to **Applications > Applications**.
   2. Click **+ Add Application**.
   3. In the **Name** field, enter a name for the application.
   4. In the **Description** field, optionally enter a description for the application.
   5. In the **Context Root** field, 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. From the **Virtual Host** list, select the virtual host you created.

      > **Note**
      >
      > The combination of virtual host and context root must be unique in PingAccess.

   7. From the **Application Type** list, select **Web**.
   8. From the **Web Session** list, select the web session you created.
   9. From the **Site** list, select the site you created that contains the application.
   10. Select the **Enabled** check box to enable the site when you save.
   11. Click **Save**.

PingAccess for Azure AD

Configure PingAccess to provide secure external access to legacy on-premises applications using PingAccess for Azure AD and Microsoft Entra ID (formerly Microsoft Azure AD).

Important

The PingAccess for Azure AD program ends on December 31, 2025. To continue using PingAccess, you must upgrade to a commercial PingAccess license. Learn more in:

- PingAccess for Azure AD Overview for an overview of license differences
- Manage license keys
- View or upload a new license

In this scenario, PingAccess provides an external path to legacy on-premises applications using the Entra ID Application Proxy through the use of header based authentication. Additionally, Microsoft Entra ID acts as the token provider for associated sessions.

PingAccess for Azure AD is a limited, free version of PingAccess for Microsoft Entra ID customers that provides protection for up to 20 applications.

This solution requires you to perform the following tasks:

- Configure PingAccess to use Microsoft Entra ID as the token provider
- Configure a PingAccess application for each application you want to protect and make available to Microsoft Entra ID as part of this solution. Applications require the configuration of:
  - A virtual host
  - A web session
  - An identity mapping
  - A site
  - An application

After you complete the configuration, you can test the application using the home page URL that you create in Microsoft Entra ID.

Get started with PingAccess for Azure AD

Protect legacy on-premises applications using Microsoft Entra ID (formerly Microsoft Azure AD) and a limited version of PingAccess called PingAccess for Azure AD.

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 Entra ID 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 through Microsoft Entra ID using the Entra ID 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.

Configuring PingAccess to use Microsoft Entra ID as the token provider

Configure PingAccess to use Microsoft Entra ID (formerly Microsoft Azure AD) as the token provider.

Before you begin

- Install PingAccess and verify that you can access the administrative console. You can find more information about installing PingAccess in Installing and Uninstalling 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, select the HTTP Proxy in the **System > Clustering** section. Learn more in Configuring administrative nodes.

About this task

You can find more information about configuring the token provider in Token provider.

Steps

1. Click **Settings**, then go to **System > Token Provider > Common > OpenID Connect**.
    1. Go to **Settings > System > Token Provider** and select **Common Token Provider**.
2. In the **Issuer** field, enter the Microsoft Entra ID **Directory ID**.

   To obtain the directory ID from Microsoft Entra ID, in the Microsoft Entra ID directory, go to **Manage > Properties** and copy the **Directory ID** value.
3. From the **Trusted Certificate Group** list,

   Choose from:

   - **Java Trust Store**
   - **Trust Any**
4. Click **Save**.

Next steps

To get the most out of the solution, see Configuring token provider-specific options.

Configuring PingAccess applications for Microsoft Entra ID

Configure PingAccess applications so they are accessible to users through the Microsoft Entra ID (formerly Microsoft Azure AD) MyApps portal.

Before you begin

- Install PingAccess and verify that you can access the administrative console. Learn more about installing PingAccess in Installing and Uninstalling PingAccess.

  > **Note**
  >
  > The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:
  >
  > ```
  > Username: Administrator
  > Password: 2Access
  > ```

- Have a Microsoft Entra ID Premium account for access to the Application Proxy feature.
- Configure Microsoft Entra ID. You can find steps to configure Microsoft Entra ID in https://docs.microsoft.com/azure/active-directory/application-proxy-ping-access.
- Configure PingAccess to use Microsoft Entra ID as the token provider.

About this task

For each application that you want to configure:

Steps

1. Create a virtual host.

   Learn more about creating a virtual host in Creating new virtual hosts.

   > **Important**
   >
   > In a typical configuration for this solution, you will create a virtual host for every application.

   1. Click **Applications**, then go to **Applications > Virtual Hosts**.
   2. Click **+ Add Virtual Host**.
   3. In the **Host** field, enter the FQDN portion of the Microsoft Entra ID **External URL**.

      Example:

      For example, external URLs of https://app-tenant.msappproxy.net/ and https://app-tenant.msappproxy.net/AppName will both have a **Host** entry of `app-tenant.msappproxy.net`.
   4. In the **Port** field, enter `443`.
   5. Click **Save**.

2. Create a web session.

   Learn more about creating a web session in Creating web sessions.

   1. Click **Access**, then go to **Web Sessions > Web Sessions**.
   2. Click **+ Add Web Session**.
   3. In the **Name** field, enter a name for the web session.
   4. From the **Cookie Type** list, select your cookie type, either **Signed JWT** or **Encrypted JWT**.
   5. In the **Audience** field, enter a unique value.
   6. In the **Client ID** field, enter the Microsoft Entra ID application ID.
   7. From the **Client Credentials Type** list, select **Secret**.
   8. In the **Client Secret** field, enter the client secret you generated for the application in Microsoft Entra ID.
   9. **Optional:** To create and use custom claims with the Microsoft Entra ID GraphAPI, click **Advanced** and clear the **Request Profile** and **Refresh User Attributes** checkboxes.

      Learn more about using custom claims in Optional - Use a custom claim.
   10. Click **Save**.

3. Create an identity mapping.

   Learn more about creating an identity mapping in Creating header identity mappings.

> **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. Click **Access**, then go to **Identity Mappings > Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. In the **Name** field, enter a name.
4. From the **Type** list, select **Header Identity Mapping**.
5. In the **Attribute to Header Mapping** table, specify the required mappings.

   Example:

   For example:

Attribute Name	Header Name
upn	x-userprincipalname
email	x-email
oid	x-oid
scp	x-scope
amr	x-amr

6. Click **Save**.

4. Create a site.

   Learn more about creating a site in Adding sites.

> **Note**
>
> In some configurations, a site might contain more than one application. A site can be used with more than one application, where appropriate.

1. Click **Applications**, then go to **Sites > Sites**.
2. Click **+ Add Site**.
3. In the **Name** field, enter a name for the site.
4. In the **Target** field, specify the 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. From the **Secure** list, select whether or not the target is expecting secure connections.
6. Click **Save**.

5. Create an application in PingAccess for each application in Microsoft Entra ID that you want to protect.

   Learn more about creating an application in Adding an application.

   1. Click **Applications**, then go to **Applications > Applications**.
   2. Click **+ Add Application**.
   3. In the **Name** field, enter a name for the application.
   4. In the **Description** field, enter a description for the application.
   5. In the **Context Root** field, 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. From the **Virtual Host** list, select the virtual host you created.

      > **Note**
      >
      > The combination of virtual host and context root must be unique in PingAccess.

   7. From the **Application Type** list, select **Web**.
   8. From the **Web Session** list, select the web session you created.
   9. From the **Site** list, select the site you created that contains the application.
   10. From the **Web Identity Mapping** list, select the mapping you created.
   11. Select the **Enabled** checkbox to enable the site when you save.
   12. Click **Save**.

Configuring applications for dual access with PingAccess for Azure AD

Configure applications for secure access both from inside and outside the network.

Steps

1. Configure an application for secure external access using Microsoft Entra ID (formerly Microsoft Azure AD) and PingAccess for Azure AD.
2. Ensure that the application is functioning as expected by signing on using the application's external Uniform Resource Locator (URL).

   Example:

   For example, http://app-tenant.msappproxy.net/.

3. In PingAccess, create a new virtual host that maps to the PingAccess host.

   Example:

   For example, `<PingAccessServerName>:3000`.

4. Assign the new virtual host to the application in addition to the virtual host specified for Microsoft Entra ID access.
5. In Microsoft Entra ID, go to the **App Registrations** window and select the application.
6. Click **Reply URLs** and add the internal PingAccess reply URL.

   Example:

   For example, `<PingAccessServerName>:3000/pa/oidc/cb`.

   > **Note**
   >
   > If you have the **Use context root as reserved resource base path** check box enabled on your PingAccess application, enter the application's context root before the reserved application context root.
   > Using the previous example, the reply URL would be `<PingAccessServerName>:3000/myApp/pa/oidc/cb` if your application had a context root of `myApp`.

7. Save the changes and test the configuration by signing on using the application's local URL.

PingAccess Monitoring Guide

PingAccess provides a range of monitoring options, from simple heartbeat options for checking responsiveness to transaction response-time logging and resource-utilization metrics. These metrics can help you gain insight into the health and performance of your PingAccess deployment.

To help you monitor the performance of a PingAccess deployment, this guide provides:

- Suggestions for key performance metrics to monitor and means by which to monitor them
- Recommendations about resource-utilization thresholds and patterns
- Monitoring options, including logs that can be used to create Splunk dashboards

The features documented here are affected by the settings in the configuration file. For more information, see the Configuration file reference.

Liveliness and responsiveness

One of the simpler methods for monitoring the performance of a PingAccess deployment is determining whether thePingAccess server is available and responsive. To help you identify the status of a server, PingAccess provides a heartbeat request endpoint.

Heartbeat endpoint

If the PingAccess server is running, the process of sending a request to the `/pa/heartbeat.ping` endpoint (or `/`*<Application Context Root>*`/pa/heartbeat.ping` if you have the **Use context root as reserved resource base path** check box enabled on your PingAccess application) returns an `OK` browser message and an `HTTP 200` status. If the request times out or requires an extended amount of time to return, the server might be overloaded or experiencing other difficulties.

If a request requires more than two or three seconds to return, multiple factors in your PingAccess deployment might be responsible. Develop a baseline for the desired response time by testing the heartbeat endpoint of your deployment at various times. This endpoint can be useful when load balancing a cluster of PingAccess server instances. Some load balancers can alter the number of requests that are sent to a particular server based on the response code received, or the responsiveness of requests that are made to the heartbeat endpoint.

The output of the heartbeat can be modified to provide performance-related information, such as CPU and memory usage, along with response times. The following example shows the JavaScript Object Notation (JSON) data that is returned when the template is changed to show the memory, CPU, and response time in milliseconds.

The following is an example of JSON data showing memory, CPU, and response time in milliseconds:

Example

```
{"items":[{
        "response.statistics.window.seconds": "5",
        "response.statistics.count": "1",
        "response.time.statistics.90.percentile":
        "129", "response.time.statistics.mean": "129",
        "response.time.statistics.max":"129",
        "response.time.statistics.min": "129",
        "response.concurrency.statistics.90.percentile": "1",
        "response.concurrency.statistics.mean": "1",
        "response.concurrency.statistics.max": "1",
        "response.concurrency.statistics.min": "1",
                "cpu.load": "15.53",
                "total.jvm.memory": "500.695 MB",
                "free.jvm.memory": "215.339 MB",
                "used.jvm.memory": "285.356 MB",
                "total.physical.system.memory": "17.18 GB",
                "total.free.physical.system.memory": "278.45 MB",
                "total.used.physical.system.memory": "16.901 GB",
                "number.of.cpus": "8",
                "hostname": "jdasilva-r",
                "open.client.connections": "1",
                "number.of.applications": "11",
                "number.of.virtual.hosts": "6",
                "last.refresh.time": "1969-12-31T18:00:00.000Z"
        }]}
```


For more information, see Heartbeat endpoint.

Response time logging

By default, the audit logs record the processing time for each transaction. With audit logging enabled, you can identify the speed with which the PingAccess server processes web and application programming interface (API) application transactions. Depending on your logging configuration, audit logging might not log any transactions. For more information, see Security audit logging.

The following example shows a default audit log with the following information:

- Total roundtrip
- Proxy roundtrip
- Userinfo roundtrip

Example

The following code example shows the processing times in milliseconds, highlighted in bold:

```
2019-12-15T17:23:12,192|GRmozOujPDDFct8RbtnfJw|tid:wUu9F0vDd9pZPKe4Oc5Ym_-RFCc..
9r72.v8c0Y2CUA5qSpvcxKHgd7QoCp|
 81 ms\| 50 ms\| 0 ms| servapp.ext.wal-ping.com [] /SimpleWebApi /:3000| joe| Cookie| 127.0.0.1| GET| /
SimpleWebApi/web/web.jsp| 200| | | Web-API| Root Resource| /
```

Resource metrics

PingAccess provides monitoring capabilities for resource-utilization metrics, such as thresholds and patterns, to strengthen the health and performance of your deployment.

PingAccess provides the following mechanisms for obtaining resource metrics:

- Java Management Extensions (JMX) - Ping recommends using JMX MBeans because this method provides a more comprehensive set of resource metric counters for analyzing performance. Several tools are available for collecting and analyzing data from JMX MBeans, including many security information and event management (SIEM) tools, such as Splunk.
- Heartbeat endpoint - For more information about enabling heartbeat message reporting, see Heartbeat endpoint.

Monitoring discusses the JConsole monitoring tool which is included with the Java SE platform. For more information about the Comprehensive JConsole, see Troubleshoot with the JConsole Tool in the Oracle Java Development Kit (JDK) documentation and The Java Monitoring and Management Console (jconsole) in the OpenJDK documentation.

Connecting with JMX

The Java Management Extensions (JMX) MBeans agent included on the Java SE platform enables connections to local and remote Java clients to monitor performance.

JConsole permits connections to local and remote Java processes.

> **Note**
>
> If your instance of PingFederate is running as a Windows service, you must connect through the remote option.

- For information on connecting to a local process, see Connecting to a local process.
- For information on connecting to a remote process, see Connecting to a remote process.

Connecting to a local process

Use the local process option to establish a connection when the PingAccess server is running on a local system.

About this task

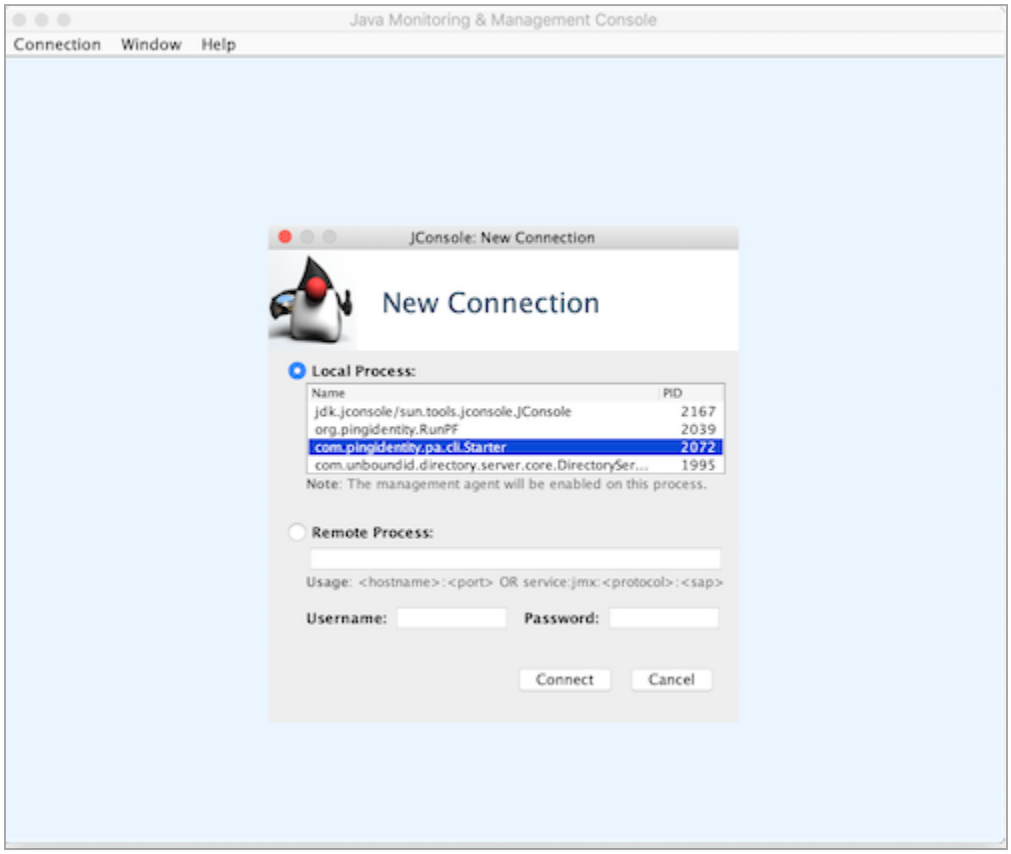
Unless you are running the PingAccess server as a Windows service, the easiest method to launch JConsole on the same machine as the server is to select **Local Process**. For information about connecting to a remote process instead, see Connecting to a remote process.

To connect to a local instance and start the monitoring process:

Steps

- In the **Local Process** list, select `com.pingidentity.pa.cli.Starter`, then click **Connect.**

> **Note**
>
> If you are running the process locally, the system might prompt you to accept the connection as insecure.



Connecting to a remote process

Use the remote process option to establish a connection when the PingAccess server is running as a Windows service, or if the `com.pingidentity.pa.cli.Starter` class is unavailable in the **Local Process** list.

About this task

Use these instructions to configure the remote process option to establish a connection. For demonstration purposes, the following task uses an Lightweight Directory Access Protocol (LDAP) configuration.

> **Note**
>
> No direct configuration support is provided for enabling remote access Java Management Extensions (JMX) for the PingAccess server. To enable this level of access, use the built-in options that are available through the Java Virtual Machine (JVM). For more information, see Monitoring and Management Using JMX Technology in the Oracle Java Development Kit (JDK) documentation.

Steps

1. In the `jvm-memory.options` file for the PingAccess server, add the following text at the end of the last memory settings:

```
#Settings to enable remote access to JMX
-Dcom.sun.management.jmxremote.port=5000"
-Dcom.sun.management.jmxremote.login.config=ExampleCompanyConfig"
#Configuration is assumed to be in the conf folder, relative path used
-Djava.security.auth.login.config=conf/ldap.config"
-Dcom.sun.management.jmxremote.ssl=false"
```

Note

Each entry must reside on its own line. In this example, a relative path is used for the `ldap.config` file. Some deployments might require a full path.

Tip

In a production environment, use Secure Sockets Layer (SSL), as shown in this example for initial testing and debugging. For information about setting up SSL, see Monitoring and Management Using JMX Technology in the Oracle JDK documentation.

2. Create the `ldap.config` file.

```
ExampleCompanyConfig {
    com.sun.security.auth.module.LdapLoginModule REQUIRED
    userProvider="ldaps://ldap.server:port/OU=where,OU=users,OU=located"
    userFilter="(&(uid={USERNAME})(objectClass=inetOrgPerson))"
    authIdentity="uid={USERNAME},OU=where,OU=users,OU=located"
    authzIdentity=monitorRole
    useSSL=true;
    };
```

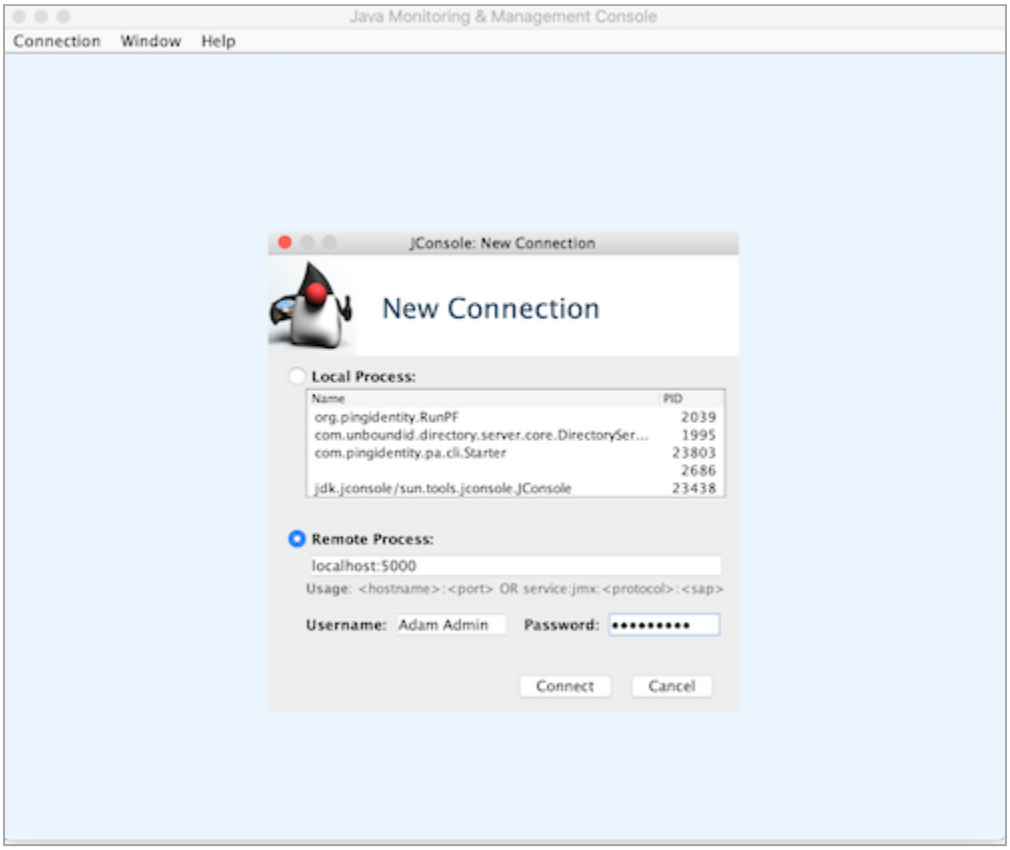
Note

Each entry must reside on its own line. In this example, `ldap.config` is placed in the PingAccess `conf` folder. If your JVM setup trusts the certificates, you can use SSL. Because of the `authIdentity` option, the configuration binds as the user that you enter. Otherwise, an anonymous bind validates the user name but not the password.

3. Place the `ldap.config` file that you created in step 2 in a location from which the PingAccess process can read it at start up.
4. If you have a clustered PingAccess environment:
    1. Perform steps 1 - 3 to each node in the cluster.
    2. Restart each node.
5. After you enable the JMX service, connect to the remote JMX service by specifying one of the following:

    Choose from:

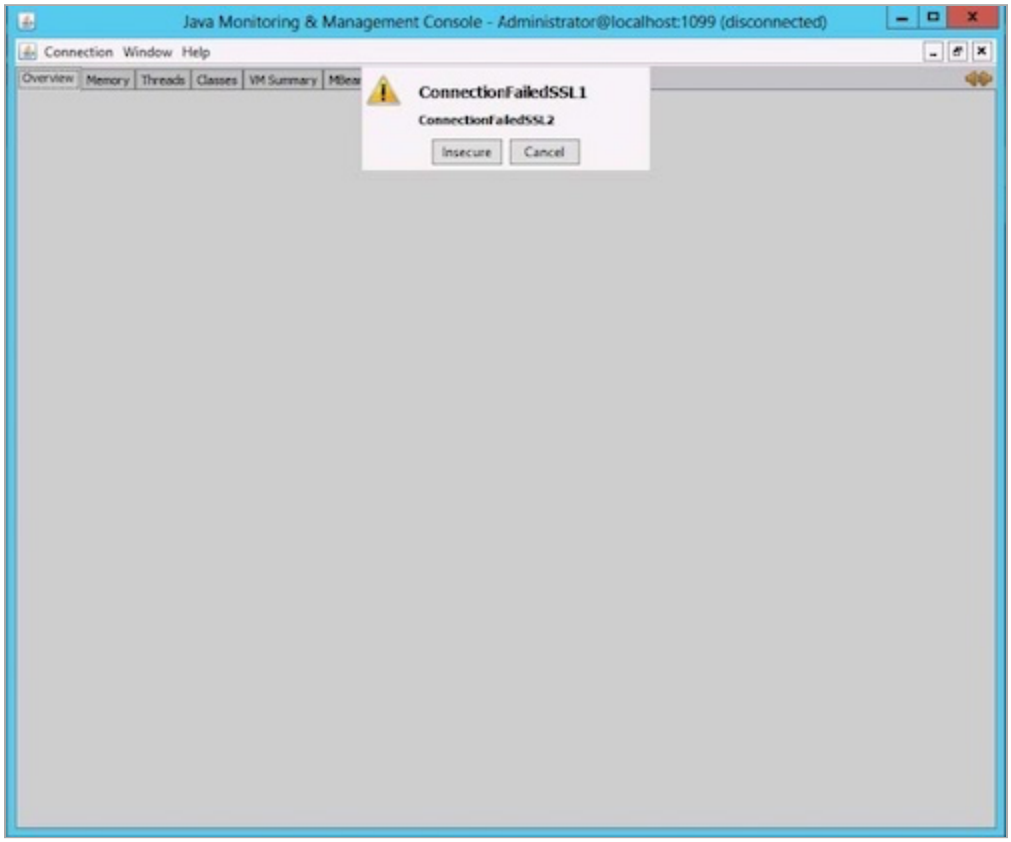
    - The name of the PingAccess server instance
    - The Internet Protocol (IP) address, port, and authentication credentials.



Note

Because JMX uses SSL by default when communicating with a remote host, the client host must trust the SSL certificate that is presented during setup for JMX. If the JMX client does not trust the JMX certificate, it displays the following message:

```
ConnectionFailedSSL1
```



Troubleshooting:

1. If SSL is enabled, import the JMX SSL certificate to the client's trusted certificates.
2. If SSL is disabled, click **Insecure** to connect.

Monitoring

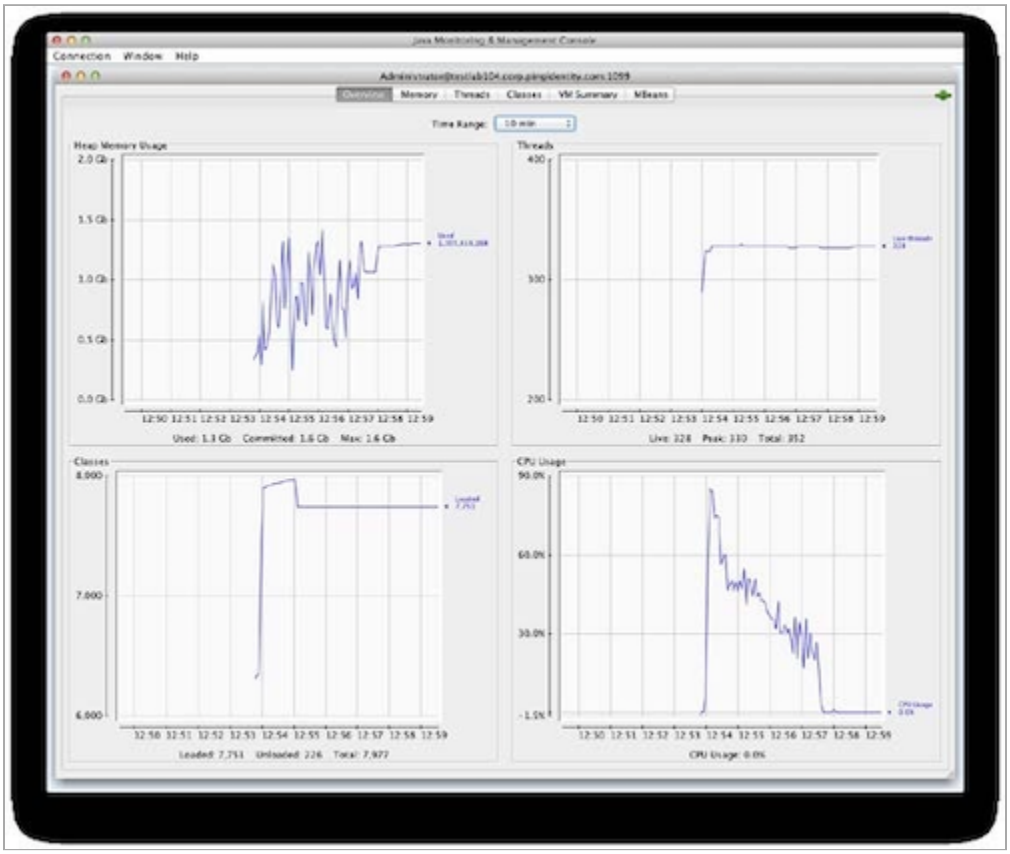
The JConsole monitoring interface is accessible after establishing a connection. This section outlines the key Java Virtual Machine (JVM) performance metrics for evaluating the activity of your PingAccess deployment.

The JConsole can be connected to multiple processes. To monitor several instances of the PingAccess server after a connection is established, go to **Connection → New Connection** and add the additional connection.

The **Overview** tab provides a dashboard of the following performance and resource-utilization charts:

- Heap memory usage (cumulative memory that is used by all memory pools)
- Live threads
- CPU usage
- Classes (number of classes that are loaded)

This tab provides a high-level view of the JVM's performance metrics.



Use the **Overview** tab to visualize and collect CPU usage data. When your PingAccess deployment is subjected to its normal or expected load, the CPU utilization typically falls between 60% and 80%. If the system registers consistently at 80% or higher, additional CPU resources might be necessary to handle load spikes that occur during peak usage times.

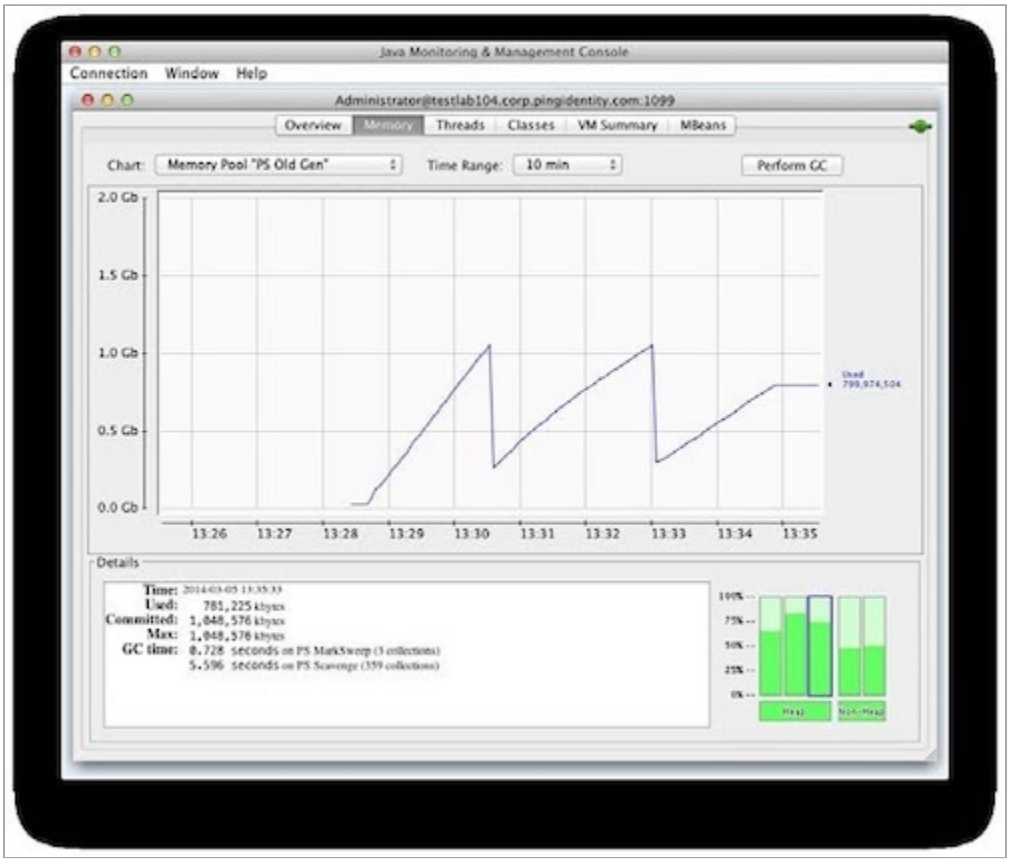
The **Overview** tab shows only overall heap usage. To view additional details about memory utilization, click the **Memory** tab, which lets you analyze usage patterns usage in specific memory pools within the heap. This tab also provides information about the overall heap utilization profile.

Objects that survive a sufficient number of garbage-collection cycles are promoted to the Old Generation. To view the memory usage in the pool of such objects, go to **Memory Pool → PS Old Gen** or **Memory Pool → G1 Old**, depending on the relevant garbage collector. The PingAccess server services mostly short-lived transactions, such as single sign-on (SSO), Security Token Service (STS), and OAuth requests. Most of the created memory objects are required only for a short period of time.

Although the PingAccess server makes use of some memory objects that are medium- to long-lived, such as session data for authentication sessions, adapter sessions, or single logout (SLO) functionality, most of the objects that are promoted to the Old Generation are likely to become garbage that requires cleaning up. If the younger generation, or Eden space, is not sized appropriately, objects are moved to and retained in the Old Generation before they are collected as garbage. If size limitations prevent the Old Generation from accumulating future garbage as well as longer-lived objects, then garbage-collection cycles occur more frequently.

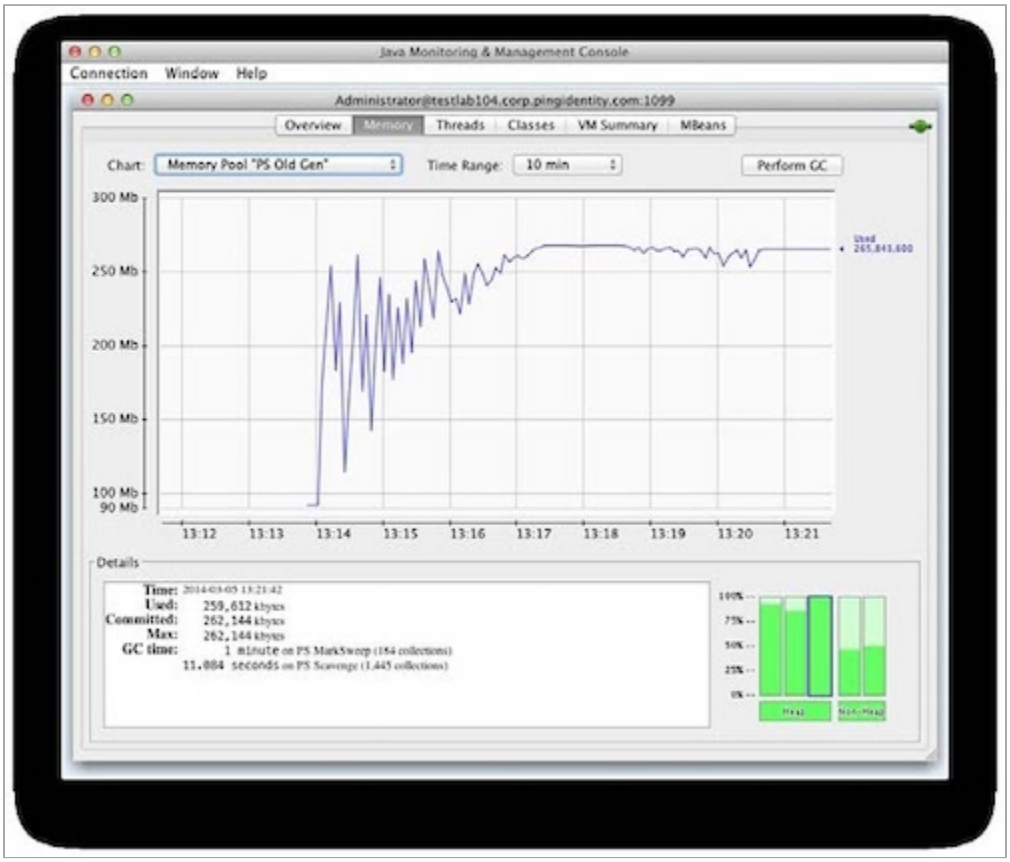
The Old Generation space is the most important space to monitor. It is easy to identify if the heap is sized and proportioned appropriately for a specific load, based on its usage pattern. The following examples involve two Old Generation usage charts. In both examples, the same user load executes the same workflow. The size of the heap represents the only difference.

Because the heap is sized adequately in the first example, memory in the Old Generation rises at a reasonably slow rate. Garbage collection frees around 60% to 75% of the space, and room is available to accommodate the future garbage of newly created objects that are moved from the Eden space, as well as the longer-term objects that remain in use. Although the space is 1 GB in size, the average full (PS MarkSweep or G1 Old Generation) collection time is approximately only 240 milliseconds, or 0.728 seconds for three collections.



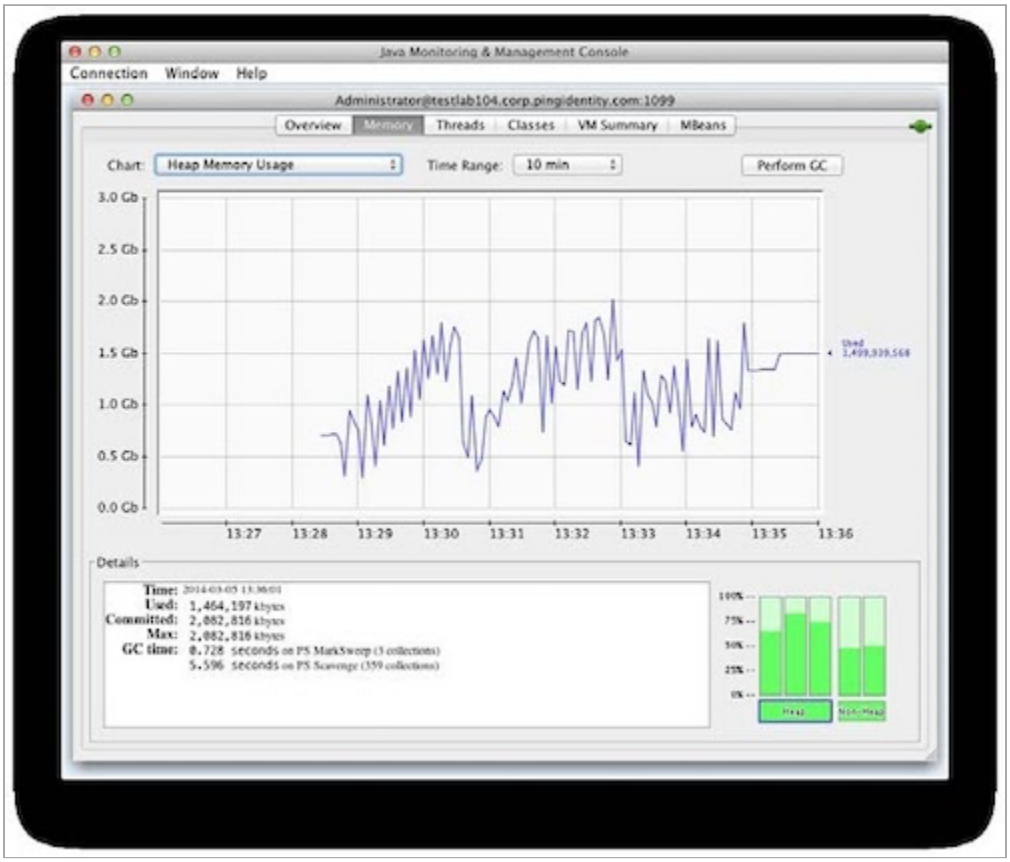
When a heap is sized inadequately, the Old Generation runs out of space.

In the following example, the amount of memory that becomes free with each garbage collection shrinks, due to the rate at which objects are promoted from the Eden space.

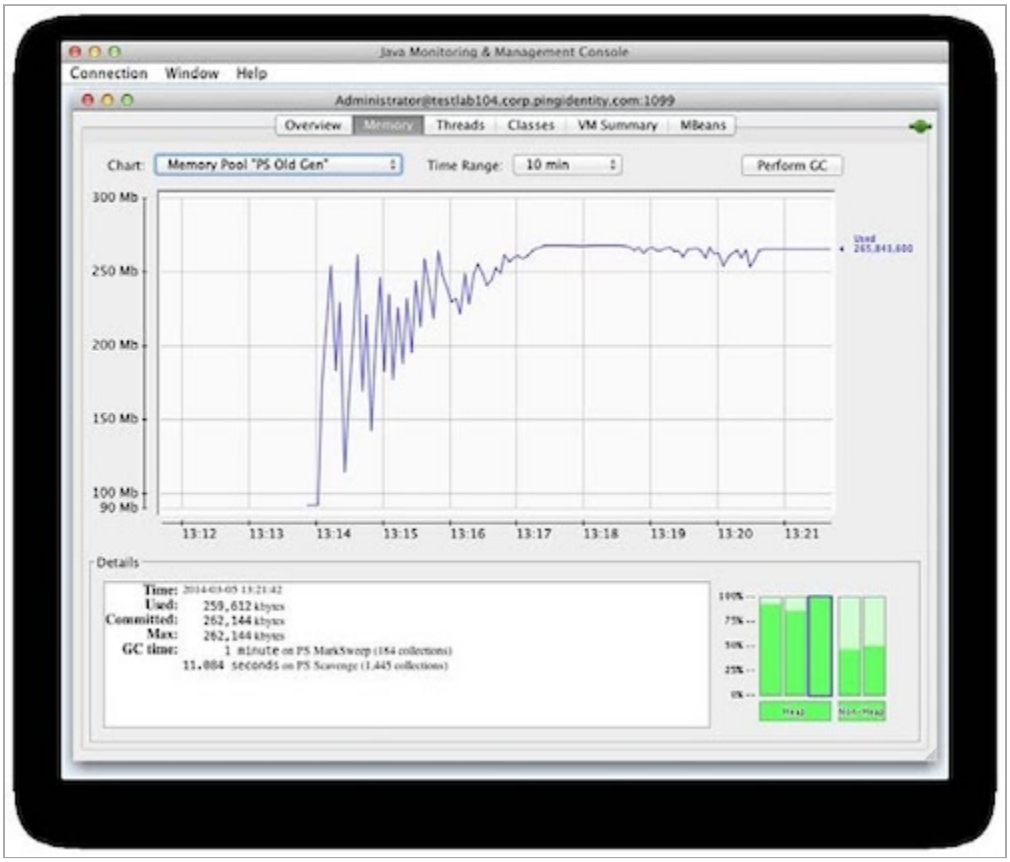


184 PS MarkSweep (full) collections require garbage collections more frequently, totaling 60 seconds, or an average of 326 milliseconds per collection.

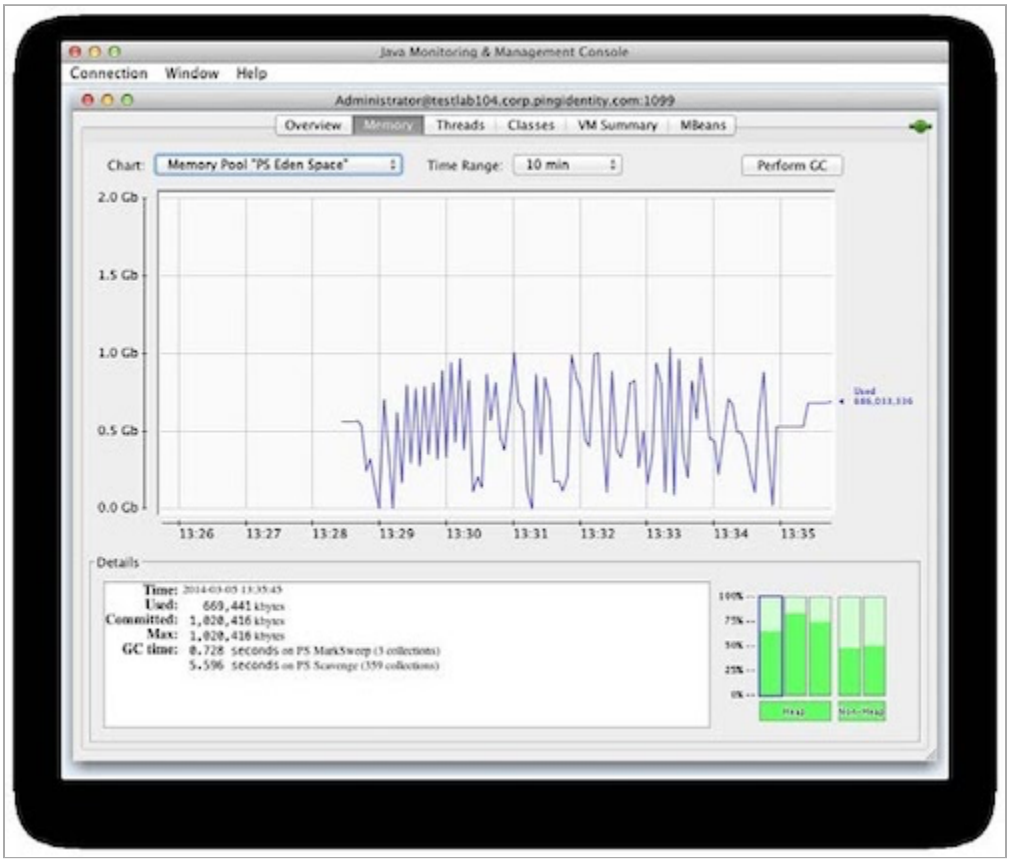
If the heap is sized appropriately for the load that the system must handle, it fills up and is followed by an appreciable drop in usage as a full garbage collection occurs, such as a PS MarkSweep collection triggered by the Old Generation filling up. In this example, the heap rises steadily, with drops from minor collections until a PS MarkSweep collection occurs and collects approximately 70% of the heap.

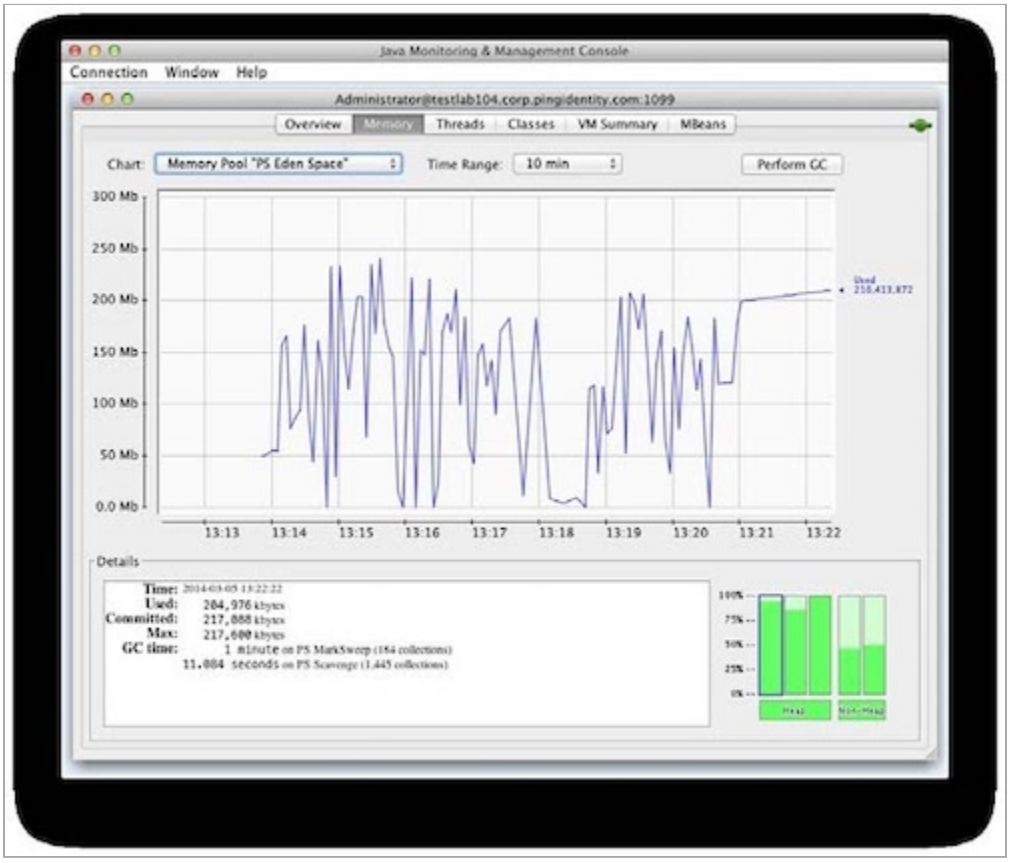


When the heap is undersized, full collections that are performed more frequently return less memory. In the following example, the frequency of Java Management Extensions (JMX) data that the JConsole retrieves does not keep pace with the frequency of full collections. As a result, only a fraction of them occur.



Regardless of whether the heap is adequately sized or undersized, the usage pattern is nearly identical with the Eden space. This similarity can be due to the sampling frequency of the data-collection tool because the number of samples might be insufficient to show that, with an undersized heap, memory is consumed and subsequently freed with greater frequency. The behavior of garbage collection in the Eden space is such that when it fills, the space is completely emptied by moving live objects to the Survivor and Old Generation spaces. Under load, the pattern resembles a jagged sawtooth, as shown in the following examples of an adequately sized heap and an undersized heap.





Because garbage collectors manage memory in the Java Runtime Environment (JRE), simply increasing the size of the heap is not always the appropriate solution. The following table outlines the total heap size recommendations for the available garbage collectors, based on available CPU resources.

For more information about garbage collectors, see Garbage collector configuration reference.

Garbage collector	Minimum recommended number of CPUs	Recommended heap size
Parallel	4	6 GB maximum
Concurrent Mark Sweep	12	4 - 6 GB minimum
Garbage First (G1)	12	6 GB minimum

If additional memory is unavailable, or if increasing the size of the heap is inadvisable because of these recommendations, the load that is handled by this instance is probably too high. In such instances, consider adding additional resources to your deployment. To verify whether the load for the instance is too high, check the CPU utilization.

To allow for the most efficient management of memory, set the minimum and maximum heap sizes to the maximum allowed values to avoid potentially expensive heap allocation resizing and divide it evenly between the young and old generations. If you are using the Garbage First collector, generational spaces are not specified through command line options because they are managed logically in real time. Even in such instances, we recommend setting the minimum and maximum heap sizes to the maximum allowed values.

For more information about fine-tuning the JVM options in the `jvm-memory.options` file, see Modifying the Java heap size in the Performance Tuning Reference Guide.

Logging, reporting, and troubleshooting

This section provides a brief summary and purpose of the available logging, reporting, and troubleshooting for PingAccess.

The following table identifies the available PingAccess logs and their purposes.

Name	Purpose
`pingaccess.log`	Primary troubleshooting log. Records PingAccess runtime and administrative server activities.
`pingaccess_engine_audit.log`	Records transactions of configured resources. Additionally, the log records transaction details when PingAccess sends requests to PingFederate, such as Security Token Service (STS), OAuth2, and JSON Web Signature (JWS) requests.
`pingaccess_api_audit.log`	Records PingAccess administrative application programming interface (API) transactions. These transactions capture activity in the PingAccess administrative console. If you're using scripts to configure PingAccess, this log also records transaction activity.

Name	Purpose
`pingaccess_agent_audit.log`	Records transactions between PingAccess agents and the PingAccess engine.
`pingaccess_sideband_client_audit.log`	Records transactions sent to and from the sideband client integration.
`pingaccess_sideband_audit.log`	Records the end-user transactions that the sideband client request captures.

- The `pingaccess.log` file is the primary troubleshooting log.
- Alongside an HTTP trace from the browser, which you can generate from a debugging application like Fiddler, the `pingaccess_engine_audit.log` and `pingaccess_agent_audit.log` files are helpful for identifying issues that must be resolved.

For more information about managing PingAccess logs, see Log configuration.

Creating an error-only server log

Modify your `log4j2.xml` file to set up a specific log to log only `ERROR`-level and higher notifications.

About this task

Monitor the `pingaccess.log` file for `ERROR`-level messages. You can configure alerts to send notifications when events occur and to improve the monitoring of these events. Even when levels are down to a minimum, the server log generates large amounts of information in an active production environment. You can set up a specific log to log only `ERROR`-level and higher alerts, which can be sent to a security information and event management (SIEM) tool, such as Splunk, when they occur.

To change your `log4j2.xml` file to enable a separate log file:

Steps

1. Create an appender.

   > **Tip**
   >
   > The simplest way to create an appender is to copy an existing one to use as a base.

   In the following example, the `RollingFile` is the same one that the `pingaccess.log` file uses. The bold text identifies items that have been changed.

```
<!-- Error Only Main Log : A size based file rolling appender -->
<RollingFile name="FILEERR"
                fileName="${sys:pa.home}/log/pingaccess.error.log"
                filePattern="${sys:pa.home}/log/pingaccess.error.log.%i"
                ignoreExceptions="false">
        <PatternLayout>
        <!-- Uncomment this if you want to use UTF-8 encoding instead of system's default encoding.
-->
        <!--
        <charset>UTF-8</charset>
        -->
        <!--
        To Activate location information uncomment the following pattern,
        comment out the current pattern and set "includeLocation" to true
        in "com.pingidentity" async logger.
        -->
        <!--
        <pattern>%d{ISO8601} %5p [%X{exchangeId}] %c:%L - %m%n</pattern>
        -->
        <pattern>%d{ISO8601} %5p [%X{exchangeId}] %c - %m%n</pattern>
        </PatternLayout>
        <Policies>
        <SizeBasedTriggeringPolicy size="100000 KB"/>
        </Policies>
        <DefaultRolloverStrategy max="10"/>
</RollingFile>
```

2. Set the appender that you created in step 1 for `AsyncRoot` at the end of your `log4j2.xml` file.

   The following example shows the necessary changes. In this example, the `level` attribute indicates the level of messages that are sent to the log file.

   ```
   <!-- Root Logger-->
   <AsyncRoot level="INFO" includeLocation="false" >
           <AppenderRef ref="File"/>
           <AppenderRef ref="FILEERR" level="ERROR"/>
   </AsyncRoot>
   ```

3. Remove the attribute `additivity="false"` from all other loggers that contain a reference to the `File` appender.

   Example:

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

   Becomes:

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

4. Restart the PingAccess server.
5. If you have a clustered environment, perform steps 1-4 on all nodes within the cluster.

> **Tip**
>
> To expedite this step, create a base file with the appropriate changes and copy it to all the nodes.

Splunk audit log

PingAccess can enable and write audit logs for Splunk to effectively collect and analyze data from Java Management Extensions (JMX) MBeans.

You can enable Splunk audit logs and use them to create dashboards in Splunk. These logs record the same information as the default audit logs, but they are formatted to facilitate parsing for specific information when you create dashboards. All of the necessary information resides within the commented-out sections.

> **Note**
>
> The link above provides instructions on how to set up the PingAccess for Splunk app, which can be found here: https://splunkbase.splunk.com/app/5368

Troubleshooting

This section covers troubleshooting for common issues with PingAccess.

- For more information on how to proceed if you're locked out of the administrative console, see Administrative SSO lockout.
- For more information on how to use the collect support data tool, see Collecting support data.
- For more information on settings that affect the size of the PingAccess cookie, see Minimizing the PingAccess cookie size.

Administrative SSO lockout

If you misconfigure administrative single sign-on (SSO) and are locked out of the PingAccess administrative console, you can disable SSO and sign on using the native sign-on.

Choose one of the following methods to disable SSO:

- If you can start the PingAccess server or the administrative node in a cluster, refer to Editing `run.properties` to disable SSO.
- If you didn't disable basic authorization, refer to Using the administrative API to disable SSO.
- If basic authorization is disabled, but administrative API OAuth is enabled, refer to Using the administrative API and a new token to disable SSO.

run.properties to disable SSO">

Editing `run.properties` to disable SSO

If you cannot sign on to the PingAccess administrative console, but you can start the PingAccess server or the administrative node in a cluster, you can edit the `run.properties` file to disable single sign-on (SSO).

About this task

If you are locked out of the administrative console because of an SSO misconfiguration, switching to basic (native) authentication makes it easier to sign on and reconfigure SSO.

Steps

1. Start the local PingAccess server.

   Learn more in Starting PingAccess.

2. Open the `run.properties` file for editing.
3. Change the `admin.auth` value from `default` to `native`.

   Example:

   ```
   admin.auth=native
   ```

4. Save your changes and restart PingAccess.

Result

You can sign on to the administrative console normally.

Next steps

Go to **Settings > Admin UI Authentication > Authentication Method** to reconfigure SSO.

Using the administrative API to disable SSO

If basic authorization wasn't disabled, use the admin application programming interface (API) to disable single sign-on (SSO).

Steps

1. Sign on to the local PingAccess system and start a non-Internet Explorer (IE) browser.
2. Sign on to the API doc page at https://*<host>*:*<admin-port>*/pa-admin-api/v3/api-docs/.

   Example:

   https://localhost:9000/pa-admin-api/v3/api-docs/

   Use the normal administrator username, `Administrator`, and your password.
3. Click and expand the **Auth** section, then expand **PUT /auth/oidc**.
4. Enter the following code:

   ```
   {
   "enabled": false
   }
   ```
5. Click **Try it Out**.

Result

You can sign on normally and reconfigure SSO for the admin API from **Settings → Admin UI Authentication → Authentication Method**.

Using the administrative API and a new token to disable SSO

If basic authorization is disabled but administrative application programming interface (API) OAuth is enabled, you can also use the administrative API to disable single sign-on (SSO).

Steps

1. Retrieve a valid token for admin API OAuth from your token provider.
2. Submit a PUT request to `https://<pa-host>/pa-admin-api/<api version>/auth/oidc` with the valid access token, where *<pa-host>* is the `hostname:port` for the PingAccess admin node and *<api-version>* is the API version (`v3` on PingAccess 5.0 or later, and `v2` on 4.X).

   The request body must contain:

```
{
 "enabled": false,
}
```

Result

You can sign on normally and reconfigure the SSO from **Settings → Admin UI Authentication → Authentication Method**.

Collecting support data

When troubleshooting, Ping Identity Support might ask you to use the collect support data tool to compile information about your PingAccess installation.

About this task

By default, the tool collects information from:

- `<PA_Home>/bin`
- `<PA_Home>/log` (the most recent files of each type within a size limit)
- `<PA_Home>/conf` (configuration files)

Information the tool collects includes:

- Environment details, such as:
  - Files present and their sizes
  - Certificate data
  - Version data
  - Java Virtual Machine (JVM) details
- System details, such as:
  - Crontab
  - Ifconfig
  - Netstat
  - Uname

> **Note**
>
> Sytem details vary depending on the operating system you're using.

The tool consists of the following files in the PingAccess home directory:

- `bin/collect-support-data.bat`
- `bin/collect-support-data.sh`

- `tools/csd/csd_configuration.yaml`
- `tools/csd/csd-1.1.jar`

> **Note**
>
> If Ping Identity Support needs more information about the PingAccess installation than the default configuration provides, Support might ask you to add a data collector to the tool by modifying its `csd_configuration.yaml` file.

To collect support data with the tool:

Steps

1. Using your PingAccess administrator account and a terminal, go to the `<PA_Home>/bin` directory.
2. Use one of the following commands to run the collect support data tool, depending on your operating system:

   Choose from:

   - On a Windows operating system, use **`./collect-support-data.bat`**.
   - On a Unix-based operating system, use **`./collect-support-data.sh`**.

   > **Note**
   >
   > If Ping Identity Support directs you to do so, you can use additional options with these commands. For example, you can run the command with the `--help` option.
   > For more information, see the PingFederate and PingAccess Support Data Collector knowledge base article.

   As the tool collects data, it displays its progress and any errors it detects. When it finishes collecting data, the tool places the data in a `.zip` file in the current directory. The format of the file's name is `support-data-ping-$<hostname>-r-$<timestamp>.zip`.
3. Review any errors that the tool brought up during the collection process, and any errors that've been added to the `support-data-ping-$<hostname>-r.log` log file.

   If necessary, resolve the errors and run the tool again.
4. Send the support data `.zip` file to Ping Identity Support.

Minimizing the PingAccess cookie size

Reduce the size of the PingAccess cookie if it causes problems in your environment.

About this task

Each of the following options can reduce the PingAccess cookie size. The exact reduction amount can't be precisely quantified because it's environment-dependent.

Steps

- When configuring the site, clear the **Send Token** checkbox. This minimizes the amount of information forwarded to the site itself. Learn more in Adding sites, Editing sites, and Site field descriptions.

- When configuring the web session, select the **Cache User Attributes** checkbox. This caches user information for use in policy decisions instead of including it in the cookie. Learn more in Creating web sessions and Editing and deleting web sessions.
- When Configuring web session management settings, select the simplest algorithms: `ECDSA using P-256 Curve` for the **Signing Algorithm** and `AES 128 with CBC and HMAC SHA 256` for the **Encryption Algorithm**.

> **Note**
>
> This option isn't as impactful as the other options and might not be possible depending on your environment's security needs.

- When Configuring admin UI SSO authentication, clear the **Include id_token_hint in SLO** checkbox.

> **Note**
>
> If your token provider requires the `id_token_hint` parameter to complete single logout (SLO), explore the other options to reduce cookie size instead.

- When Configuring OpenID Connect token providers, clear the **Track token_id** checkbox.

> **Note**
>
> If you want to use the `id_token` attribute in an identity mapping, rule, or virtual logout resource, explore the other options to reduce cookie size instead.

- When Configuring PingOne Advanced Identity Cloud or PingAM as the token provider, clear the **Track token_id** checkbox.

> **Note**
>
> If you want to use the `id_token` attribute in an identity mapping, rule, or virtual logout resource, explore the other options to reduce cookie size instead.