PingIntelligence

May 30, 2025

PINGINTELLIGENCE
Version: 5.2;latest

Copyright

All product technical documentation is
Ping Identity Corporation
1001 17th Street, Suite 100
Denver, CO 80202
U.S.A.

Refer to https://docs.pingidentity.com for the most current product documentation.

Trademark

Ping Identity, the Ping Identity logo, PingAccess, PingFederate, PingID, PingDirectory, PingDataGovernance, PingIntelligence, and PingOne are registered trademarks of Ping Identity Corporation ("Ping Identity"). All other trademarks or registered trademarks are the property of their respective owners.

Disclaimer

The information provided in Ping Identity product documentation is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Table of Contents

PingIntelligence for APIs

PingIntelligence for APIs uses artificial intelligence (AI) to secure APIs in your environment by identifying and automatically blocking cyberattacks on APIs, exposing active APIs, and providing detailed reporting on all API activity.

Release Notes

- PingIntelligence 5.2 (September 2023)
- PingIntelligence 5.1 (December 2021)
- PingIntelligence 5.0.1 (August 2021)
- Previous Releases

Get Started with PingIntelligence

- Introduction to PingIntelligence for APIs
- PingIntelligence Docker evaluation deployment
- PingIntelligence Docker toolkit
- Automated deployment
- Manual deployment
- PingIntelligence Kubernetes deployment

Use PingIntelligence

- Use Case: Converting SSL certificates to ASE compatible format
- PingIntelligence Integrations

Copyright © 2025 Ping Identity Corporation

Troubleshoot PingIntelligence

- PingIntelligence Health Check Guide

Learn More

- Community
- Partner portal

Copyright © 2025 Ping Identity Corporation

Introduction to PingIntelligence for APIs

PingIntelligence for APIs uses artificial intelligence (AI) to secure APIs in your environment. It identifies and automatically blocks cyberattacks on APIs, exposes active APIs, and provides detailed reporting on all API activity.

Leveraging AI models specifically tailored for application programming interface (API) security, PingIntelligence brings cyberattack protection and deep API traffic insight to existing API gateways and application server-based API environments. It detects anomalous behavior on APIs and the data and applications exposed through APIs. It also automatically blocks attacks across your API environment.

Key components

PingIntelligence is a suite of three interconnected components, API Security Enforcer (ASE), API Behavioral Security (ABS) AI engine, and PingIntelligence Dashboard:

ASE

The first processing layer in PingIntelligence. It captures the metadata of the monitored APIs and sends it to the ABS AI engine. You can deploy ASE in two modes, inline and sideband. When deployed in inline mode, ASE directly receives the API traffic or can work alongside other load balancers like AWSElastic Load Balancing (ELB). In sideband mode, ASE integrates with API gateways in an ecosystem and extends the cybersecurity of PingIntelligence.

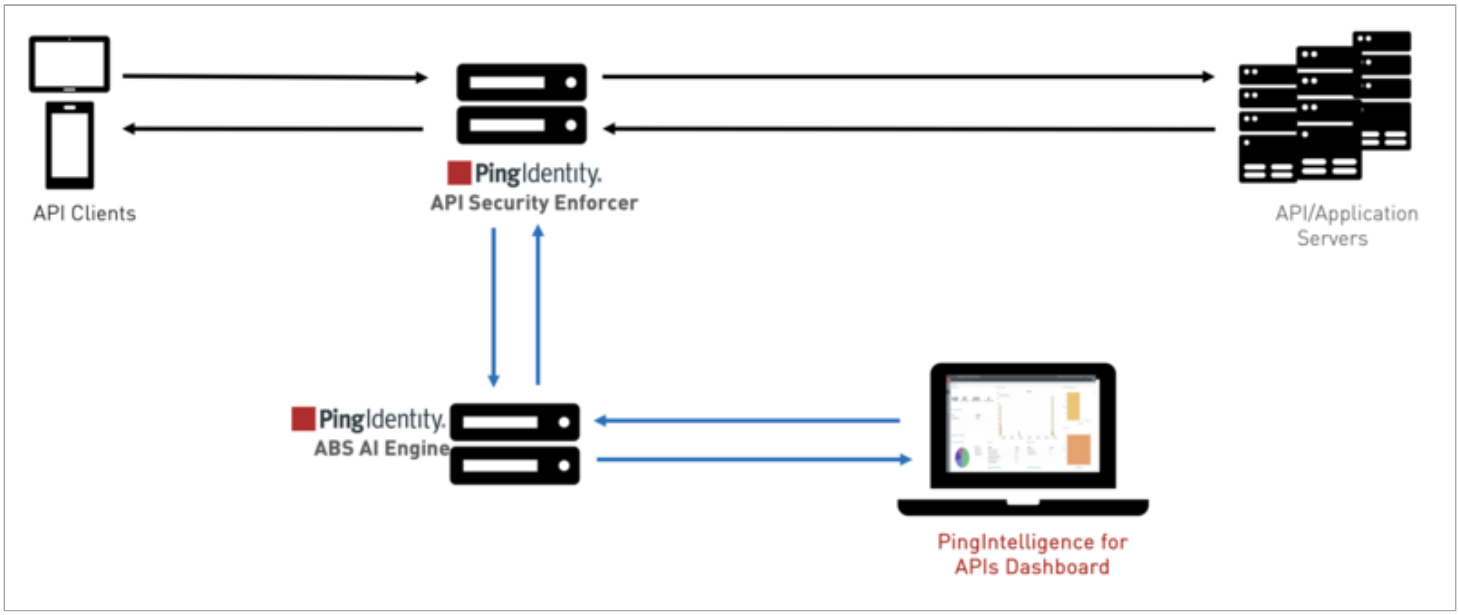
ABS AI engine

The AI engine infers the API traffic patterns in the metadata from ASE. It builds machine learning models that self-train based on the API traffic. ABS detects the attacks on APIs and blocks the clients from which the attacks originate. It also provides in-depth forensics on the activities performed by a client. The reports provide detailed information on the activity from an individual token, Internet Protocol (IP) address, cookie, API key, or username. In addition to attack detection, ABS continuously discovers the new and unknown APIs in an API ecosystem and brings them under observation.

PingIntelligence Dashboard

PingIntelligence Dashboard provides rich analytics on API activities in an environment. It gives granular insights at an API level and across APIs. It can provide information on the training status of APIs, different kinds of attacks on APIs, and much more. PingIntelligence Dashboard also supports admin activities such as attack management and discovery of APIs.

Copyright © 2025 Ping Identity Corporation



Related links

- Sideband ASE
- Inline ASE
- ABS AI Engine
- PingIntelligence for APIs Dashboard

Copyright © 2025 Ping Identity Corporation

Release Notes

New features and improvements in PingIntelligence for APIs. Updated September 28, 2023.

PingIntelligence 5.2 (September 2023)

PingIntelligence for APIs 5.2 provides the following enhancements:

New in Dashboard

Improved

The PingIntelligence for APIs Dashboard is enhanced to provide an improved user experience with the following functionalities:

Enhanced Main Dashboard

The main dashboard adds tiles with quick links for **Discovered APIs**, **API Count**, and **Indicators of Attack**. See PingIntelligence Dashboard.

Enhanced SIEM integration

The security information and event management (SIEM) integration provides a webhooks connection to a Splunk SIEM. The SIEM integration also enables a customer to combine anomalous API activity data with events from other security tools.

New in AI Engine

Improved

Support for detection of user-based broken object-level authorization (BOLA), broken function-level authorization (BFLA), user-based data injection, and anomalous token claim detection. PingIntelligence detects and optionally blocks these manipulations and malicious activity. For more information, see Indicators of attack.

PingIntelligence 5.1 (December 2021)

PingIntelligence for APIs 5.1 provides the following enhancements:

New in Dashboard

Improved

The PingIntelligence for APIs Dashboard is enhanced to provide an improved user experience for the following functionalities:

- New **PingOne Dashboard** provides a streamlined user interface with support for drill down into API details, blocklisted clients, and clients flagged for Indicators of Attack (IoAs). The rearchitected Dashboard significantly accelerates the processing of API metadata to speed updates to administrators on API activity and abnormal events. See Dashboard.

Copyright © 2025 Ping Identity Corporation

- An updated **Attack management GUI** delivers more detailed information to assist security administrators in analyzing Indicators of Attack (IoAs). The enhanced reporting includes additional insight into why a client's behavior was flagged, suggested remediation steps, and transaction-level details from API requests and responses associated with the anomalous behavior. See Attack management.
- **Enhanced SIEM integration** pushes the same detailed IoA information (e.g. why flagged, remediation steps, transaction data) available via the Attack Management GUI to a SIEM. The SIEM integration enables a customer to combine anomalous API activity data with events from other security tools.
- **Automated Publishing of Discovered APIs** supports distributed discovery of APIs across multiple datacenters from a centralized or cloud-based Dashboard.

New in AI Engine

Improved

Improved Anomalous API Header and Query String Detection

Updated AI algorithms detect anomalous values and content in API headers or query strings. Examples include hackers manipulating content, executing malicious scripts, passing attack variables, accessing unauthorized content, and other abnormal behavior. PingIntelligence detects and optionally blocks these manipulations and malicious activity. For more information, see IoAs (Indicators of Attack).

New in ASE

New

Real-Time Enforcement of Missing Token

For inline or sideband deployments, ASE can be configured to detect and automatically block clients not presenting a token to APIs requiring access tokens.

New Kubernetes Deployment

New

Support for production PingIntelligence deployments in AWS EKS using a Ping-supplied Helm-Chart. See PingIntelligence Kubernetes deployment.

Resolved Issue: ABS AI engine

Fixed

ABS AI engine has been updated to use a Log4j version with the fixes for the critical vulnerabilities.

Resolved Issue: Dashboard update

Fixed

Copyright © 2025 Ping Identity Corporation

Dashboard has been updated to use a Log4j version with the fixes for the critical vulnerabilities.

PingIntelligence 5.0.1 (August 2021)

PingIntelligence for APIs 5.0.1 provides the following updates:

ASE Integration

Improved

The API Security Enforcer (ASE) now supports integration with PingOne platform. You can deploy ASE on-premise in either inline or sideband mode and integrate it with PingOne. This hybrid integration delivers API security through the new PingOne API Intelligence service.

New PingIntelligence Docker Toolkit Environment Variables

New

The PingIntelligence Docker toolkit adds environment variables to support integration of ASE with the PingOne API Intelligence platform. For more information, see PingIntelligence Docker toolkit. The new environment variables support:

- Configuring gateway credentials to use for connecting ASE with the PingOne API Intelligence platform
- Setting the ABS AI engine deployment type to cloud or on-premise

PingIntelligence 5.0 (June 2021)

PingIntelligence 5.0 provides the following enhancements:

All PingIntelligence components now support a single unified license

New

PingIntelligence now supports up to 10 subpath levels for API base paths when API Security Enforcer (ASE) is deployed in sideband mode. Subpath depth is the number of sub-paths for a unique API definition. For more information, see Discovery sub-paths.

Dashboard Enhancements

New

The PingIntelligence Dashboard is enhanced to provide improved user experience for the following functionalities:

- The updated **Attack management** page gives a comprehensive view of Indicators of Attacks (IoAs) on a per client basis. A separate **Enable / Disable Attacks** page helps the administrators in efficient attack management. For more information, see Indicators of Attacks on REST APIs.

Copyright © 2025 Ping Identity Corporation

- The **Training Status** page now allows you to view the training status for an API by selecting the API from a drop-down list. The page also has a new capability to download per API and across API attack thresholds in a JSON formatted text file. For more information, see Training period status.

ASE Updates

New

ASE has the following new additions:

- **External Load Balancer support for ASE to ABS AI Engine traffic -**ASE can be optionally configured to utilize external load balancers to distribute traffic across ABS AI Engine nodes. This provides the flexibility to support auto-scale of ABS AI Engine nodes and more high availability configurations.
- **REST API for sideband token management -**The `Token API` helps to create, import, and delete ASE sideband tokens. You can also retrieve the list of tokens issued by ASE.
- **REST API for sideband authentication -** The `Authentication API` helps to enable and disable ASE sideband authentication. You can also retrieve the authentication status. For more information, see REST APIs for sideband token and authentication.

New in sideband integration policies

Improved

- **NGINX plus policy -**The updated PingIntelligence sideband policy can seamlessly integrate with NGINX Plus R22 or R23 systems. This enhanced policy supports NGINX nodes with PingAccess agents installed and can capture user information from the PingAccess token introspection. For more information, see Installing NGINX Plus for RHEL 7.6.
- **Apigee policy -**The updated PingIntelligence sideband policy adds optional asynchronous communication between Apigee and ASE for improved performance when deployed in environments that do not require automated client blocking. For more information, see Apigee integration.
- **Kong policy -** The PingIntelligence sideband policy is enhanced to support extraction of user information from JWTs when OpenID Connect (OIDC) plugin is installed in a Kong gateway. For more information, see Kong API gateway integration.

New in deployment tools

Improved

The following PingIntelligence deployment tools have been modified to support integration with PingOne:

- PingIntelligence Ansible deployment framework
- PingIntelligence Docker PoC deployment
- PingIntelligence Kubernetes deployment
- PingIntelligence Docker toolkit

Copyright © 2025 Ping Identity Corporation

Previous Releases

This page shows changes and updates to PingIntelligence for APIs.

- PingIntelligence 4.4 (December 2020)
    - PingIntelligence 4.4.1 (April 2021)
- PingIntelligence 4.3
- PingIntelligence 4.2
- PingIntelligence 4.1
    - PingIntelligence 4.1.0
    - PingIntelligence 4.1.1

Copyright © 2025 Ping Identity Corporation

Installing PingIntelligence for APIs

PingIntelligence Docker evaluation deployment

This guide describes the installation and execution of PingIntelligence for APIs software in a Docker environment for inline and sideband deployment.

Important

- The setup requires the Community Version (CE) of Docker 18.09 or later. Make sure that the Docker infrastructure is set up before proceeding with installation and setup of PingIntelligence for APIs software.
- The Docker images provided are only for PingIntelligence for APIs evaluation purposes. Do not use the Docker images in production deployments or for setting up environments for security testing.

The automation script imports and installs the Docker images. A script is run to generate normal application programming interface (API) traffic to train the artificial ingelligence (AI) engine. After training is complete, another script is run to send a mixture of normal and attack traffic. Then, you can learn how to access a graphical dashboard that shows activity on the test environment and detailed reporting on the API activity.

This Docker Evaluation Guide provides instructions for deploying a test configuration. The Docker setup can be deployed in an inline mode where the client traffic directly reaches API Security Enforcer (ASE). It can also be deployed in sideband mode where the API traffic reaches the API gateway and the API gateway sends the request to ASE. For more information on sideband and inline deployment, see Sideband ASE and Inline ASE.

Installation requirements

Review the following software, documentation, and server requirements.

Docker images

Note

The setup requires the Community Version (CE) of Docker 18.09 or later. Make sure that the Docker infrastructure is set up before proceeding with installation and setup of the PingIntelligence for APIs software.

Download the Docker evaluation package. The Docker package creates the following five containers on the host machine:

1. API Security Enforcer (ASE)
2. API Behavioral Security Engine (ABS)
3. PingIntelligence for APIs Dashboard
4. Client that sends the traffic

Copyright © 2025 Ping Identity Corporation

5. Google Go App server

PingIntelligence license

You must have a PingIntelligence for APIs license to start ASE and ABS. Contact the Ping Identity sales team to access the trial license.

Postman reporting

ABS generates various REST application programming interface (API) reports. You can view these reports using the Postman client or any other REST API client. PingIntelligence provides a Postman collection to view the various ABS reports.

Download the Postman client from the Postman site.

Documentation

- For information about administering ASE, inline ASE, real-time cybersecurity, and so on, see the ASE Admin Guide.
- For more information about administering ABS, artificial intelligence (AI) engine training, various REST API reports, and so on, see the ABS Admin Guide.
- For information about how to access and use the Dashboard, see PingIntelligence Dashboard.

Server requirements

The setup requires one machine that hosts the six Docker images. The server requirements for the machine are specified in the following table.

OS	Ubuntu 18.04 LTS
Hardware	8 CPUs, 32 GB RAM, 500 GB Storage

> **Note**
>
> The server requirement is for a single server for evaluation purposes only.

Downloading and untarring the Docker package

Download and untar the Docker package to install the PingIntelligence license.

Before you begin

Contact the Ping Identity sales team for instructions on accessing the Docker package and acquiring a valid license for PingIntelligence API Security Enforcer (ASE) and API Behavioral Security (ABS).

About this task

> **Note**
>
> By default the Docker evaluation package is configured for deployment in inline mode. If you want to deploy Docker in sideband mode, see Configuring Docker evaluation sideband deployment.

Copyright © 2025 Ping Identity Corporation

When you have access to the Docker package and a valid PingIntelligence license:

Steps

1. Download and save the Docker package in the `/opt` directory.
2. Untar the package by running the following command:

   ```
   $sudo tar -xf /opt/pi-api-docker-poc-5.1.tar.gz
   ```

3. Change the directory to `/opt/pingidentity/docker-poc`.
4. Copy the license file in the `pingidentity/docker-poc/license` directory and make sure that the license file is named `PingIntelligence.lic`.

   Example:

   The following is a sample of the license file:

   ```
   ID=
   Organization=
   Product=
   Version=
   IssueDate=
   EnforcementType=
   ExpirationDate=
   MaxTransactionsPerMonth=
   Tier=
   ```

Configuring Docker evaluation sideband deployment

Configure the Docker package for sideband deployment.

Before you begin

Download PingIntelligence sideband policies and documentation from the Ping Identity Downloads site.

About this task

You can optionally configure the Docker evaluation environment for a sideband deployment with an application programming interface (API) Gateway. The Docker evaluation package ships with sample API swagger definition files that can be adapted to support your API Gateway environment.

To configure the Docker package for sideband deployment:

Steps

- Open the `config` directory and edit the `poc.config` file to set `mode` as `sideband`.

  Example:

  The following is a sample `poc.config` file:

Copyright © 2025 Ping Identity Corporation

```
# API Security Enforcer mode.
# allowed values: inline, sideband
ase_mode=inline

# initial training period in hours
training_period=1

# poc mode for training
poc_mode=true


 Below Configuration is applicable only when ase_mode is set to sideband


# API gateway ip address or dns name
gateway_ip=
# API gateway port
gateway_port=443
# set gateway protocol if API gateway is configured with ssl
# else set it to tcp
# allowed values: tcp, ssl
gateway_protocol=ssl
```

The following table describes the parameters.

Parameter	Description
`ase_mode`	Defines the deployment mode of ASE. Possible values are `inline` and `sideband`. The default value is `inline`.
`training_period`	Training period of AI engine in hours. The minimum value is 1 hour.
`poc_mode`	Defines the mode in which ABS artificial intelligence (AI) engine trains its models. The default value is `true`. **Note** You should keep the value as `true`. If you change it to `false`, it could take longer to set all the attack thresholds.
`gateway_ip`	Configure the Uniform Resource Locator (URL) for API gateway.
`gateway_port`	Port number of API gateway URL
`gateway_protocol`	API gateway protocol. Possible values are `ssl` or `tcp`.

Copyright © 2025 Ping Identity Corporation

Installing and loading Docker images

Install and load Docker images from the `images` directory.

About this task

To install and load Docker images:

Steps

- Run the following command on the host Ubuntu 18.04 machine:

```
/opt/pingidentity/docker-poc$sudo ./bin/start.sh install
```

Example:

```
sudo ./bin/start.sh install
Tue Dec 28 01:59:50 MST 2021 : loading ASE image
Loaded image: pingidentity/ase:5.1
Tue Dec 28 01:59:54 MST 2021 : loading ABS image
Loaded image: pingidentity/abs:5.1
Tue Dec 28 02:00:03 MST 2021 : loading API Publish image
Loaded image: pingidentity/apipublish:5.1
Tue Dec 28 02:00:11 MST 2021 : loading Dashboard image
Loaded image: pingidentity/dashboard:5.1
Tue Dec 28 02:00:43 MST 2021 : loading mongo image
Loaded image: pingidentity/mongo:4.2.0
Tue Dec 28 02:00:50 MST 2021 : loading Kafka image
Loaded image: pingidentity/kafka:5.1
Tue Dec 28 02:01:07 MST 2021 : loading Zookeeper image
Loaded image: pingidentity/zookeeper:5.1
Tue Dec 28 02:01:18 MST 2021 : loading client image
Loaded image: pingidentity/client:5.1
Tue Dec 28 02:01:25 MST 2021 : loading server image
Loaded image: pingidentity/server:5.1
Tue Dec 28 02:01:32 MST 2021 : Installation completed successfully
```

Setting up the Docker evaluation environment

Start and set up the Docker evaluation environment.

About this task

To start and set up the Docker containers:

Steps

1. Run the following command on the host Ubuntu machine:

```
/opt/pingidentity/docker-poc$sudo ./bin/start.sh setup
```

Result:

Copyright © 2025 Ping Identity Corporation

```
sudo ./bin/start.sh setup
Tue Dec 28 02:03:14 MST 2021 : Starting setup scripts
ASE is running in inline mode and POC mode is set to true
Training period configured: 1 hour(s)
Creating network pingidentity_net
Creating service pingidentity_mongo
Creating service pingidentity_dashboard
Creating service pingidentity_kafka
Creating service pingidentity_server
Creating service pingidentity_ase
Creating service pingidentity_abs
Creating service pingidentity_client
Creating service pingidentity_apipublish
Creating service pingidentity_zookeeper
Tue Dec 28 02:04:47 MST 2021 : Setup successful
```

2. Wait for a minute after the successful completion of the setup and run the following command to verify that API Security Enforcer (ASE) and API Behavioral Security (ABS) have started:

```
#sudo docker service logs pingidentity_ase | grep 'API Security Enforcer started'
#sudo docker service logs pingidentity_abs | grep 'ABS started'
```

Troubleshooting:

If a wrong license is installed, you receive the following error message:

```
Fri Jun 18 06:32:27 UTC 2021 : Starting setup scripts
License not found. Please place PingIntelligence.lic file at /<pi-inistallpath>/pingidentity/docker-
poc/license/ location
```

Next steps

> **Note**
>
> If the PingIntelligence for APIs Dashboard is configured with `SSO` mode, then update the contents of the `cert/webgui-sso-oidc-provider.crt` file with the PingFederate public certificate.

Starting the training

The PingIntelligence for APIs artificial intelligence (AI) engine needs to be trained before it can start detecting attacks on your APIs.

About this task

The training duration is 85 minutes.

To start the training:

Steps

- Run the following command: `/opt/pingidentity/docker-poc$sudo ./bin/start.sh training`

Copyright © 2025 Ping Identity Corporation

Result:

```
root@vortex-108:/opt/pingidentity/docker-inline-poc$sudo ./bin/start.sh training
Tue Mar 31 06:44:25 UTC 2020 : Starting model training scripts
Tue Mar 31 06:44:25 UTC 2020 : Model training started. Wait 1 hr 25 minutes for the model training
to complete.
```

Generating sample attacks

Generate sample attacks.

About this task

To generate sample attacks on the preconfigured APIs:

Steps

- Run the following command:

```
/opt/pingidentity/docker-poc$sudo ./bin/start.sh attack
```

Result:

```
root@vortex-108:/opt/pingidentity/docker-poc$sudo ./bin/start.sh attack
Tue Mar 31 09:13:02 UTC 2019 : Starting attack scripts
Tue Mar 31 09:13:02 UTC 2019 : Attack started.
```

Viewing deception APIs

View the deception APIs.

About this task

The deception application programming interface (API) is part of the Docker setup. The deception command completes the following steps:

- Enables API Security Enforcer (ASE) detected attacks.
- Fetches the list of configured APIs from ASE.
- Sends traffic to the decoy API and receives a `200 OK` response.
- Sends traffic to a regular API (for example, `shopapi`). The connection is blocked because any client that previously accessed a decoy API is not allowed access to production APIs.

> **Note**
>
> API deception works only for inline Docker evaluation setup.

Copyright © 2025 Ping Identity Corporation

Steps

- Run the following script to test API deception:

```
root@vortex-108:/opt/pingidentity/docker-poc$sudo./bin/start.sh deception
```

Example:

Copyright © 2025 Ping Identity Corporation

```
Enabling enable_ase_detected_attack on ASE...
Press any key to continue
ASE Detected Attack is now enabled
Fetching the list of APIs from ASE
Press any key to continue
decoy ( loaded ), http, decoy: out-context, client_spike_threshold: 0/second,
server_connection_queueing: disabled
shop-books ( loaded ), http, client_spike_threshold: 300/second, server_connection_queueing:
disabled
shop-electronics ( loaded ), http, decoy: in-context, client_spike_threshold: 700/second,
server_connection_queueing: enabled
shop ( loaded ), http, decoy: in-context, client_spike_threshold: 300/second,
server_connection_queueing: disabled
Sending traffic to "decoy API" with client IP 10.10.10.10...
Press any key to continue
curl -v http://localhost:8000/decoy/myhome -H "X-Forwarded-For: 10.10.10.10"
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8000 (#0)
> GET /decoy/myhome HTTP/1.1
> Host: localhost:8000
> User-Agent: curl/7.47.0
> Accept: /
> X-Forwarded-For: 10.10.10.10
>
< HTTP/1.1 200 OK
< Server: ASE
< Content-Length: 2
< Connection: close
<
* Closing connection 0
OK
Accessing regular API using client IP 10.10.10.10...
Press any key to continue
curl -v http://localhost:8000/shopapi/login -H "Host: shopapi" -H "Content-Type: application/text" -
H "X-Forwarded-For: 10.10.10.10" -d 'user=root'
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8000 (#0)
> POST /shopapi/login HTTP/1.1
> Host: shopapi
> User-Agent: curl/7.47.0
> Accept: /
> Content-Type: application/text
> X-Forwarded-For: 10.10.10.10
> Content-Length: 9
>
* upload completely sent off: 9 out of 9 bytes
< HTTP/1.1 401 Unauthorized
< Server: ASE
< Connection: close
< content-length: 19
<
* Closing connection 0
Error: Unauthorized
Error: Unauthorized
```


Copyright © 2025 Ping Identity Corporation

API discovery

API Behavioral Security (ABS) discovers the APIs when the discovery is enabled.

The automated setup sets up the discovery mode. APIs are discovered by ABS when a global application programming interface (API) is defined in the PingIntelligence API Security Enforcer (ASE). The discovered APIs from ABS are added to ASE. API model training starts after the APIs are added in ASE.

For more information, see API discovery and configuration.

Accessing the PingIntelligence Dashboard

Access the PingIntelligence for APIs Dashboard from a browser at the default Uniform Resource Locator (URL): https://*<pi_install_host>*:8030.

About this task

There are two preconfigured login users in PingIntelligence for APIs Dashboard:

- `admin`
- `ping_user`

Multiple users can share the `admin` and `ping_user` logins simultaneously on PingIntelligence Dashboard. The admin user has access to all PingIntelligence Dashboard functions. A `ping_user` can only view the application programming interface (API) dashboards.

The PingIntelligence Dashboard is categorized into the following components:

Dashboard Component	Description
Main Dashboard	Available for `admin` and `ping_user`
APIs	Available only for `admin` user
Discovered APIs	Available only for `admin` user
Attack Management	Available only for `admin` user
License	Available only for `admin` user
Active Sessions	Available only for `admin` user
Settings	Available only for `admin` user

> **Note**
>
> For further information on dashboard features, usage, and administration, see PingIntelligence Dashboard.

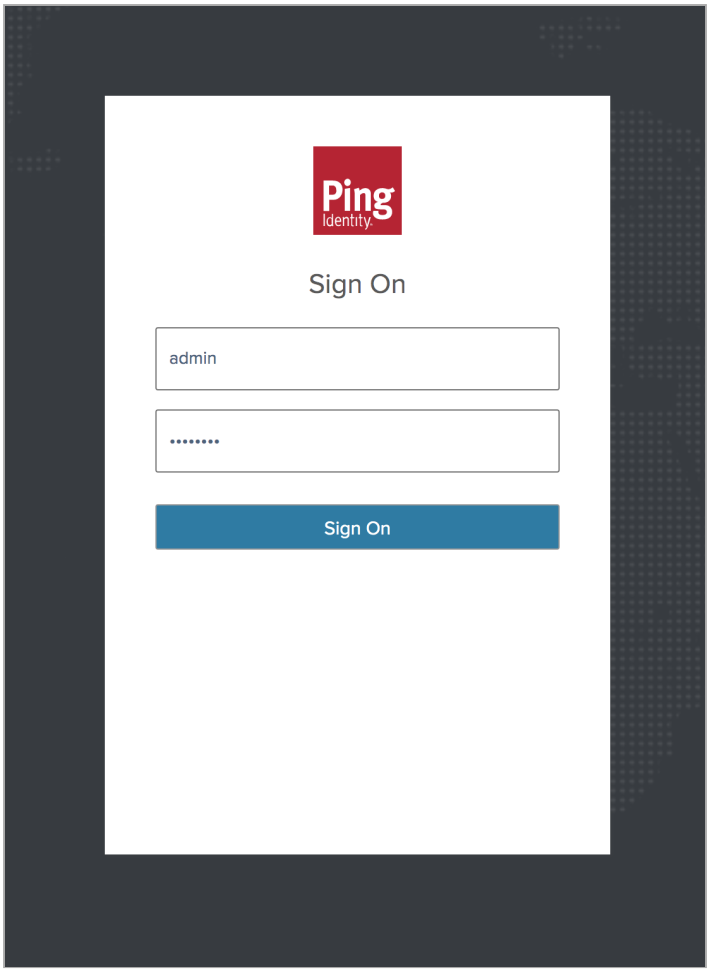
Copyright © 2025 Ping Identity Corporation

Steps

1. At the sign-on prompt, sign on as `admin` or `ping_user`.

   The default password for both the users is `changeme`.

   > **Caution**
   >
   > You must change the default password for production deployments. However, in a Docker Proof of Concept deployment, use the default password.



   > **Note**
   >
   > If the Dashboard is not accessible, check if the default port (8030) was changed by your system administrator.

2. **Optional:** Change the password using the following command-line interface (CLI) command:

```
#  <pi_install_dir>/webgui/bin/cli.sh -u admin update_ui_password --username -value  <admin or
ping_user>  --new-password -p
Enter admin password >  <current admin password>
Enter new password >  <new password>
Reenter new password >  <new password>
success: password updated.
```


Copyright © 2025 Ping Identity Corporation

3. **Optional:** To configure the maximum number of active sessions, set the `pi.webgui.session.max-active-sessions` parameter in the `<pi_install_dir>/webgui/config/webgui.properties` file.

   The default value is 50.

4. **Optional:** To delete active sessions, enter the following command:

   ```
   #  <pi_install_dir>/webgui/bin/cli.sh -u  <username>  -p  <password>  delete_sessions
   ```

   > **Note**
   >
   > You need to have admin user privileges to delete active user sessions.

   Result:

   The current active users will be prompted to sign on again to the Dashboard.

Viewing ABS detailed reporting

The API Behavioral Security (ABS) Engine REST application programming interface (API) interface provides access to a range of JavaScript Object Notation (JSON) reports on attacks, metrics, and anomalies.

About this task

To view these reports, Ping Identity provides templates that can be loaded into Postman to simplify viewing of the JSON reports.

To install and configure Postman software:

Steps

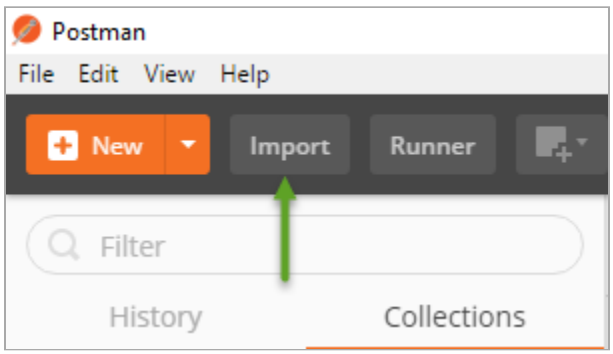
1. Download and install the Postman application 6.2.5 or later.
2. Download the API Reports Using Postman Collection from the Automated Docker Proof of Concept Installation section.

   `ABS_5.0_Environment` and `ABS_5.0_Reports` are files for Postman.

3. Launch the Postman application. Make sure to disable SSL verification in Postman.

   Learn more in Using self-signed certificate with Postman.

4. Import the downloaded reports files by clicking the **Import** button.

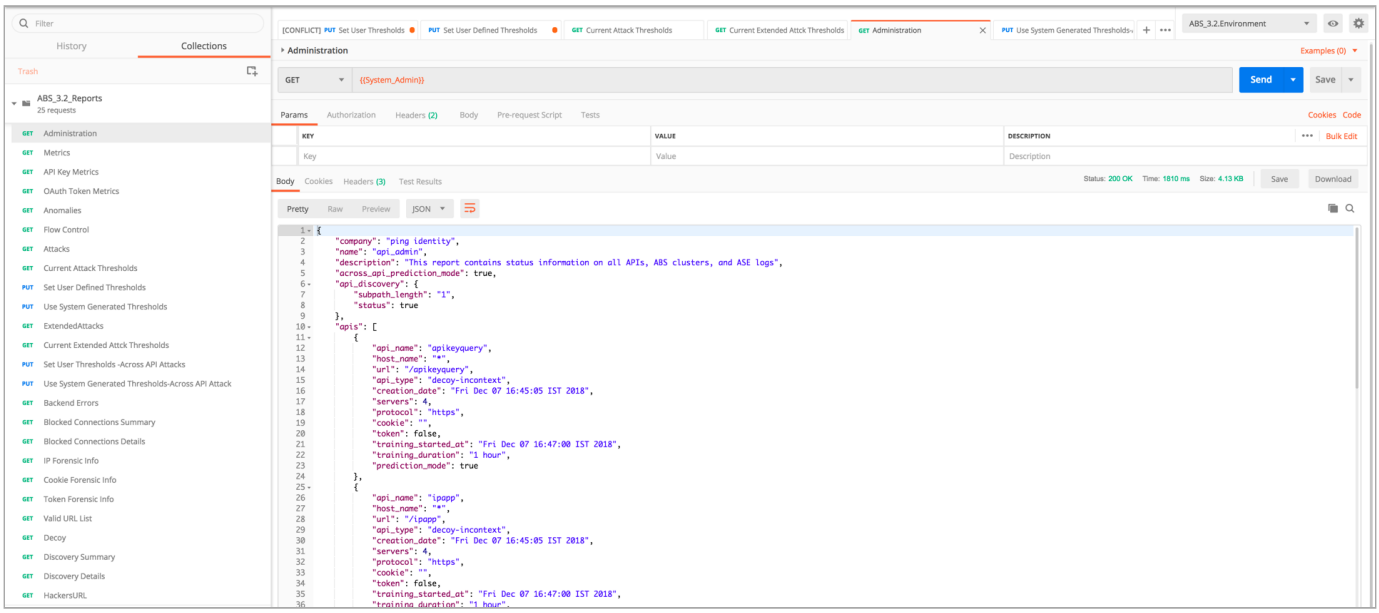


Copyright © 2025 Ping Identity Corporation

5. Click the **Gear** icon in the top-right corner.
6. In the pop-up window, click **ABS_5.1_Environment**.
7. In the **Edit Environment** pop-up window, configure the following values and click **Update**:

Value	Description
Server IP Address	IP address of the Docker machine
Port	Default is 8080
Access_Key, **Secret_Key**	Default **Access_Key** is `abs_ak` and default **Secret_Key** is `abs_sk`
API_Name	The name of API to view in reports
Later_date, **Earlier_date**	A range of dates to query

8. In the main Postman app window, select the report to display in the left column and then click **Send**.



Next steps

Other reports that can be generated for a specified timeframe include the following. Make sure to specify a time range that covers the time that you ran the attack scripts.

Report	Description
Metrics	Shows all activity on the specified API.

Copyright © 2025 Ping Identity Corporation

Report	Description
Attacks	(set Type=0) Shows a list of all attack categories and client identifiers (for example, token, IP address, cookie) associated with the attack.
Backend Errors	Shows activity that generated the errors.
IP Forensic Info	Set the IP address to an attacker identified in the Attacks report. Shows all API activity for the specified IP.
Token Forensic Info	Set the token address to an attacker identified in the Attacks report. Shows all API activity for the specified token.

Shutting down the Docker evaluation environment

Shut down the Docker evaluation environment.

About this task

You can stop the Docker evaluation environment to delete all containers and the data.

Steps

- Run the following command:

```
sudo ./bin/stop.sh
```

Example:

```
Tue Dec 28 02:02:43 MST 2021 : Starting stop scripts
Removing service pingidentity_abs
Removing service pingidentity_apipublish
Removing service pingidentity_ase
Removing service pingidentity_client
Removing service pingidentity_dashboard
Removing service pingidentity_kafka
Removing service pingidentity_mongo
Removing service pingidentity_server
Removing service pingidentity_zookeeper
Removing config pingidentity_webgui_sso_oidc_provider_crt
Removing network pingidentity_net
```

Verifying the Docker evaluation setup

Verify the Docker evaluation setup.

About this task

Copyright © 2025 Ping Identity Corporation

The PingIntelligence products are installed in the `/opt/pingidentity` directory within the Docker container.

To verify the setup:

Steps

1. List all the containers with the **`docker ps`** command.
2. To get console access for any of the Docker containers, fetch the Container ID of the Docker container using the **`docker ps`** command output and use it in the following command:

   ```
   #docker exec -it  <docker-container-id>  /bin/bash
   ```

3. To get the service names of the containers, run the following command:

   ```
   #docker service ls
   ```

   The service name is the second column in the output.

4. To check the log of any service, run the following command:

   ```
   #docker service logs  <service name>
   ```

   Example:

   The following example shows what this command might look like in your system:

   ```
   # docker service logs pingidentity_ase
   ```

PingIntelligence Docker toolkit

PingIntelligence for APIs provides a Docker toolkit to create Docker images of PingIntelligence components and MongoDB.

You can run the Docker toolkit on either RHEL or Ubuntu machines. The supported versions are RHEL 7.9 or Ubuntu 18.0.4 LTS.

The Docker toolkit provides information on environment variables available for the PingIntelligence components and an example Kubernetes `.yaml` file for automated deployment of PingIntelligence in Kubernetes environments.

For more information on using the `.yaml` file, see PingIntelligence Kubernetes deployment.

Untarring the Docker toolkit

To use the Docker toolkit, you need to untar the toolkit.

Before you begin

Copyright © 2025 Ping Identity Corporation

You must:

- Download the following PingIntelligence components, tools, and open source modules:
  - PingIntelligence API Security Enforcer (ASE) 5.0
  - PingIntelligence API Behavioral Security (ABS) 5.0
  - PingIntelligence Dashboard 5.0
  - MongoDB 4.2.0
  - OpenJDK 11.0.2 to 11.0.6
  - Kibana 6.8.1
  - Elasticsearch 6.8.1
- Obtain valid PingIntelligence for APIs license files from the Ping Identity Sales team.

> **Note**
>
> - Download the correct ASE binary based on the base image you want to create.
> - Download the correct MongoDB 4.2.0 binary based on the Docker image you want to build.

About this task

To untar the Docker toolkit:

Steps

1. To untar the toolkit, run the following command:

```
tar -zxf pi-api-docker-toolkit-5.1.tar.gz
```

Result:

Untarring the Docker toolkit creates the directory structure as shown in the following table.

Directory	Description
`bin`	Contains the **`build.sh`** script to build the Docker images.
`config`	Contains the `docker.conf` file to configure the base image name and the base image operating system.
`certs/webgui`	Contains the PingFederate public certificate file, `webgui-sso-oidc-provider.crt`. The PingIntelligence Dashboard Docker image can be generated by optionally packaging it with the PingFederate public certificate.

Copyright © 2025 Ping Identity Corporation

Directory	Description
`certs/`	Contains the folders `\{ase, abs, apipublish, dataengine, webgui, kafka, mongo, elasticsearch}`. These contain certificate and key files for PingIntelligence components. The keystore will be generated during image creation with the password configured in `docker.conf`. **Note** The PingIntelligence Dashboard has the following components: ◦ `dataengine` ◦ `webgui` You can configure separate certificates and keys for each of them. However, the keystore password will be common.
`data`	For internal use.
`docker-toolkit`	For internal use.
`external`	Contains the third-party software: ◦ MongoDB 4.2.0 ◦ Elasticsearch 7.13.4 ◦ OpenJDK 11.0.2 to 11.0.6
`helm-chart`	For internal use.
`images`	Contains the Docker images created using the **`build.sh`** script.
`keystore`	For internal use.
`lib`	For internal use.
`license`	Contains the PingIntelligence license file. **Note** You can build the images without adding the license file to the `license` directory. If you build the Docker images without the license file in `license` directory, then you need to map or mount the license file in the `/config/` directory.
`logs`	Contains the log files.
`software`	Contains PingIntelligence ASE, ABS, and Dashboard.

2. To configure `docker.conf`, navigate to the `config` directory and edit the `docker.conf` file for base image name and base image operating system.

   Example:

   The following is a sample `docker.conf` field:

Copyright © 2025 Ping Identity Corporation

```
# Base image name using which all the PingIntelligence images are created
base_image=registry.access.redhat.com/rhel7:7.9

# Operating system of the base image. The valid values are ubuntu or rhel
base_image_os=rhel

# Define the username for images. This user is added to the Docker
# images. Containers created from these Docker images use the configured # user to run
PingIntelligence software
user_name=pinguser

# Define the username for images. This user is added to the Docker
# images. Containers created from these Docker images use the configured # user to run
PingIntelligence software
group_name=pinggroup

#Define keystore password for different component
#These will be used to create keystore while building images through crt and key while.ASE keystore
password can be changed from helm values.
abs_keystore_password=changeme
apipublish_keystore_password=changeme
dashboard_keystore_password=changeme
kafka_keystore_password=changeme
```

Note

- The setup requires the Community Version (CE) of Docker 18.09 or later.
- Do not set the `user_name` as `root` in the `docker.conf` file.

Building the PingIntelligence Docker images

Use the **`build.sh`** script available in the `bin` directory to build the Docker images.

About this task

You can build all the following Docker images at once, or you can choose to build the images individually. The following Docker images are built:

- API Security Enforcer (ASE)
- API Behavioral Security (ABS)
- Dashboard
- MongoDB
- API Publish
- Kafka
- Zookeeper

Copyright © 2025 Ping Identity Corporation

> **Tip**
>
> You should obfuscate the various keys and password in ASE, ABS, and the Dashboard before building the Docker images.

For more information on obfuscating keys and passwords, see the following topics:

- ASE - Obfuscating keys and passwords
- ABS - Obfuscate passwords
- Dashboard - Obfuscate keys and passwords
- API Publish - Obfuscating passwords

To build the Docker images:

Steps

1. Configure the base image name and base image operating system details in the `config/docker.conf` file.
2. Download the following PingIntelligence software to the `software` directory:
    1. ASE
    2. ABS
    3. PingIntelligence Dashboard
3. Download OpenJDK 11.0.2, Elasticsearch 7.13.4, Kafka 2.5.0, Zookeeper 3.5.7, and MongoDB 4.2.0 in the `external` directory and save them with their respective names shown in the following table.

    > **Note**
    >
    > Make sure that MongoDB is as per the base image configured in the `docker.conf` file. Always build Kafka and Zookeeper images first, if building individually.

Software	File Name
Elasticsearch	`elasticsearch.tar.gz`
OpenJDK 11.0.2	`openjdk11.tar.gz`
Kafka	`kafka.tar.gz` Download the Kafka tar version 2.5.0 tar from https://archive.apache.org/dist/kafka/2.5.0/kafka_2.12-2.5.0.tgz
Zookeeper	`zookeeper.tar.gz` Download the Zookeeper tar version 3.5.7 tar from https://archive.apache.org/dist/zookeeper/zookeeper-3.5.7/apache-zookeeper-3.5.7-bin.tar.gz

Copyright © 2025 Ping Identity Corporation

Software	File Name
MongoDB	`mongodb.tgz`

4. Run the **`build.sh`** script to build the Docker images:

```
docker-setup# ./bin/build.sh all
Base image os: rhel
Creating build context for ASE
Creating Image
Image created with tag pingidentity/ase:5.0
Image saved to /home/ubuntu/docker-setup/images/pingidentity_ase.tar
Creating build context for abs
Creating Image
Image created with tag pingidentity/abs:5.0
Image saved to /home/ubuntu/docker-setup/images/pingidentity_abs.tar
Creating build context for dashboard
Creating Image
Image created with tag pingidentity/dashboard:5.0
Image saved to /home/ubuntu/docker-setup/images/pingidentity_dashboard.tar
Creating build context for mongo
Creating Image
Image created with tag pingidentity/mongo:4.2.0
Image saved to /home/ubuntu/docker-setup/images/pingidentity_mongo.tar
root@ip-172-31-25-146:/home/ubuntu/docker-setup# vim lib/dashboard/context/entrypoint.sh
Creating build context for apipublish
Creating Image
Image created with tag pingidentity/apipublish:5.1
Image saved to /home/ubuntu/docker-setup/images/pingidentity_apipublish.tar
Creating build context for kafka
Creating Image
Image created with tag pingidentity/kafka:5.1
Image saved to /home/ubuntu/docker-setup/images/pingidentity_kafka.tar
Creating build context for zookeeper
Creating Image
Image created with tag pingidentity/zookeeper:5.1
Image saved to /home/ubuntu/docker-setup/images/pingidentity_zookeeper.tar
```

The other options that you can give with **`build.sh`** are:

- `ase`
- `abs`
- `dashboard`
- `mongo`
- `apipublish`
- `kafka`
- `zookeeper`

5. Verify that the images are created by checking the local registry. Run the following command:

Copyright © 2025 Ping Identity Corporation

```
sudo docker image ls | grep pingidentity
```

Example:

```
pingidentity/dashboard              5.1       e9bbbb21c14d   18 hours ago    1.69GB
pingidentity/apipublish             5.1       bc3878f8a340   23 hours ago    777MB
pingidentity/mongo                  4.2.0     21a8177c0a35   23 hours ago    640MB
pingidentity/abs                    5.1       540c5c384fba   23 hours ago    766MB
pingidentity/ase                    5.1       262d9207d5de   23 hours ago    424MB
pingidentity/zookeeper              5.1       1cf24526f8cf   23 hours ago    690MB
pingidentity/kafka                  5.1       f9118757f234   23 hours ago    737MB
```

6. Verify that the Docker images are saved in the `images` directory:

```
docker-setup# ls -ltra images/
```

> **Note**
>
> The Docker images do not install any additional packages like `vi` editor and so on.

Example:

```
-rw-------.  1 root root  601075200 Dec  2 00:24 pingidentity_kafka.tar
-rw-------.  1 root root  554516992 Dec  2 00:24 pingidentity_zookeeper.tar
-rw-------.  1 root root  287269376 Dec  2 00:24 pingidentity_ase.tar
-rw-------.  1 root root  628353024 Dec  2 00:24 pingidentity_abs.tar
-rw-------.  1 root root 1549656576 Dec  2 00:25 pingidentity_dashboard.tar
-rw-------.  1 root root  503818752 Dec  2 00:26 pingidentity_mongo.tar
-rw-------.  1 root root  637405184 Dec  2 00:26 pingidentity_apipublish.tar
```

Environment variables exposed in Docker images

Environment variables are exposed in the Docker images.

If you do not set the environment variable, the default values are used. The following tables list the environment variables for API Security Enforcer (ASE), API Behavioral Security (ABS), Dashboard, and MongoDB.

ASE Environment Variables

The following table lists the ASE environment variables and the values.

Environment	Value	Usage
`MODE`	`inline/sideband`	ASE can be deployed either in inline mode or sideband mode. For more information, see the ASE admin guide.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`TIMEZONE`	`string`	Set the timezone of ASE to either `local` or `UTC`. The default value is `utc`. **Note** Make sure `TIMEZONE` is set to the same value in ASE, ABS, and Dashboard.
`ENABLE_CLUSTER`	`true/false`	Set the value to `true` to enable ASE cluster.
`ENABLE_ABS`	`true/false`	Set the value to `true` to enable ABS.
`PEER_NODE`	`<IP or hostname>`:port	ASE cluster peer node's IP address and port number.
`ASE_SECRET_KEY`	`string`	Set the value of the ASE secret key. **Note** The ASE access key cannot be changed. Its value always remains `admin`.
`ABS_ENDPOINT`	`<IP or hostname>`:port	IP address or host name of the ABS endpoint.
`ABS_ACCESS_KEY`	`string`	Access key to connect to ABS.
`ABS_SECRET_KEY`	`string`	Secret key to connect to ABS.
`ADMIN_LOG_LEVEL`	`1-5`	1-5 (FATAL, ERROR, WARNING, INFO, DEBUG)
`ENABLE_SIDEBAND_AUTHENTICATION`	`true/false`	Enable client side authentication. This setting is applicable only in sideband mode. When enabled, ASE authenticates requests using ASE authentication tokens.
`ENABLE_SIDEBAND_KEEPALIVE`	`true/false`	Set the value to `true` to enable connection keepalive for requests from gateway to ASE. This configuration is applicable only in `sideband` mode.
`ENABLE_ASE_HEALTH`	`true/false`	Set the value to `true` to enable ASE health check module.
`ENABLE_GOOGLE_PUBSUB`	`true/false`	Google Pub/Sub configuration.
`GOOGLE_PUBSUB_TOPIC`	`string`	Google Pub/Sub topic.
`GOOGLE_PUBSUB_CONCURRENCY`	`number`	The number of concurrent connections to Google Pub/Sub. Minimum: 1, Default: 1000, Maximum: 1024
`GOOGLE_PUBSUB_QPS`	`number`	The number of messages published per second. Minimum: 1, Default: 1000, Maximum: 10000

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`GOOGLE_PUBSUB_APIKEY`	`string`	Google service account API key (Optional)
`CACHE_QUEUE_SIZE`	`number`	The maximum number of messages buffered in memory. If the queue is full, messages are written to `logs/google_pubsub_failed.log`. Minimum: 1, Default: 300, Maximum: 10000
`GOOGLE_PUBSUB_TIMEOUT`	`number`	Timeout in seconds to publish a message to Google Pub/Sub. Minimum: 10, Default: 30, Maximum: 300
`DEPLOYMENT_TYPE`	`string`	Indicates ABS deployment type to ASE. Supported values are `onprem` or `cloud`.
`GATEWAY_CREDENTIAL`	`string`	The obfuscated gateway credentials that are generated at cloud portal. ASE parses these gateway credentials to get OAuth Uniform Resource Locator (URL) and URL for ABS API calls. Populate this value when `DEPLOYMENT_TYPE` is set to `cloud`.
`ENABLE_ABS_PUBLISH`	`true/false`	Set this value to `true` to allow ASE to fetch the published API list from ABS.
`ABS_PUBLISH_REQUEST_MINUTES`		This value determines how often ASE will get the published API list from ABS.
`ENABLE_STRICT_REQUEST_PARSER`	`true/false`	Enable strict parsing checks for client requests. • `true`: ASE will block requests with an invalid header start. • `false`: ASE will allow requests.

ABS Environment Variables

The following table lists the ABS environment variables and the values.

Environment	Value	Usage
`MONGO_RS`	`mongodb://<IP or hostname>:<port>,<IP or hostname>:<port>, <IP or hostname>:<port>`	MongoDB replica set IP addresses or host names and port numbers.
`MONGO_USERNAME`	`string`	MongoDB username.
`MONGO_PASSWORD`	`string`	MongoDB password.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`ABS_LOG_LEVEL`	`string`	Log levels (`ALL > DEBUG > INFO > WARN > ERROR > FATAL > OFF`) The default is `INFO` .
`MONGO_SSL`	`true/false`	Set to `true` if MongoDB instance is configured in SSL mode. By default, ABS will try to connect to MongoDB using non-SSL connection. The default is `false` .
`IS_DASHBOARD_NODE`	`true/false`	Setting as true makes an ABS node for dashboard engine query only and does not participate in ABS cluster for log processing.
`ENABLE_EMAILS`	`true/false`	Enable (`true`) or disable (`false`) ABS email notifications.
`SENDER_EMAIL`	`string`	The email address used for sending email alerts and reports.
`SENDER_EMAIL_PASSWORD`	`string`	The password of the sender's email account. **Note** You can leave this field blank if your SMTP server does not require authentication.
`RECEIVER_EMAIL`	`string`	The email address notified about alerts and reports. If you want more than one person to be notified, use an email alias.
`ABS_CLI_ADMIN_PASSWORD`	`string`	Set the ABS command-line interface (CLI) admin password.
`ABS_JKS_PASSWORD`	`string`	Set the ABS Java keystore password.
`MONGO_CERTIFICATE_VERIFY`	`true/false`	Set to true if you want to enable verification of MongoDB SSL server certificate. By default, ABS will try to connect to MongoDB without verifying SSL connection. The default is `false` .
`TIMEZONE`	`string`	Set the timezone of ABS to either `local` or `UTC` . The default value is `utc` . **Note** Make sure `TIMEZONE` is set to the same value in ASE, ABS, and the Dashboard.
`ABS_ACCESS_KEY`	`string`	The access key for the ABS admin user. For more information, see ABS users.
`ABS_SECRET_KEY`	`string`	The secret key for the ABS admin user. For more information, see ABS users.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`ABS_ACCESS_KEY_RU`	`string`	The access key for the restricted user. For more information on restricted users, see ABS users.
`ABS_SECRET_KEY_RU`	`string`	The secret key for the restricted user. For more information on restricted users, see ABS users.
`ATTACK_INITIAL_TRAINING`	`integer`	The attack training period.
`ATTACK_UPDATE_INTERVAL`	`integer`	The attack threshold uphold interval.
`API_DISCOVERY`	`true/false`	Set the value to `true` to enable application programming interface (API) discovery in ABS. For ABS to discover APIs, a global API JavaScript Object Notation (JSON) must be configured in ASE. For more information, see API discovery and configuration.
`API_DISCOVERY_INITIAL_PERIOD`	`integer`	The initial period set in hours in which ABS has to be discover APIs. It is good practice to keep the API discovery interval period less than the initial attack training interval.
`API_DISCOVERY_UPDATE_INTERVAL`	`integer`	The time period in hours in which ABS reports the newly discovered APIs.
`API_DISCOVERY_SUBPATH`	`integer`	The number of subpaths that are discovered in an API. The maximum value is 3.
`POC_MODE`	`string`	Sets the mode in which ABS trains its API models. Set it to `true` for running ABS in evaluation deployment mode. For more information, see Configuring and verifying ABS POC mode.
`KAFKA_SERVERS`	`string`	The Kafka `ip:port` needs to be configured.
`ABS_CONSUMER_USER`	`string`	ABS consumer user in Kafka.
`ABS_PRODUCER_USER`	`string`	ABS producer user in Kafka.
`ABS_CONSUMER_GROUP`	`string`	ABS group in Kafka.
`ABS_CONSUMER_PASSWORD`	`string`	ABS consumer user password.
`ABS_PRODUCER_PASSWORD`	`string`	ABS producer user password.
`KAFKA_MIN_INSYNC_REPLICA`	`integer`	The number of minimum in-sync replicas for data in Kafka.
`TRANSACTIONS_TOPIC`	`string`	ABS transaction topic in Kafka.
`ATTACK_TOPIC`	`string`	ABS attack topic in Kafka.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`ANOMALIES_TOPIC`	`string`	ABS anomalies topic in Kafka.

MongoDB Environment Variables

The following table lists the MongoDB environment variables and the values.

Environment	Value	Usage
`MONGO_USERNAME`	`string`	MongoDB username.
`MONGO_PASSWORD`	`string`	MongoDB password.
`MUTLI_NODE_REPLICA_SET`	`string`	Set it to `true` if you wan to run multiple MongoDB nodes in MongoDB replica set. The default value is `false`. If you have set to it to `true`, then manually add MongoDB nodes into replica set. Run `abs_init.js` script from the primary MongoDB node.
`WIRED_TIGER_CACHE_SIZE_GB`	`float`	Memory in GB to be used by MongoDB cache.
`MONGO_SSL`	`string`	Configures whether MongoDB uses SSL. The default value is `false`.
`MONGO_PORT`	`string`	Custom port for Mongo.

Dashboard Environment Variables

The following table lists the Dashboard environment variables and the values.

Environment	Value	Usage
`DISCOVERY_SOURCE`	`string`	Source of API discovery. Values can be `abs`, `pingaccess`, or `axway`.
`PINGACCESS_URL`	`string`	The URL of PingAccess if you set the discovery source as `pingaccess`.
`PINGACCESS_USERNAME`	`string`	The PingAccess username for API discovery.
`PINGACCESS_PASSWORD`	`string`	The PingAccess password for API discovery.
`AXWAY_URL`	`string`	The URL of Axway if you set the discovery source as `axway`.
`AXWAY_USERNAME`	`string`	The Axway username for API discovery.
`AXWAY_PASSWORD`	`string`	The Axway username for API discovery.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`DISCOVERY_MODE`	`string`	The mode in which the Dashboard publishes APIs to ASE. Values can be `auto` or `manual`. For more information, see Discovered APIs.
`DISCOVERY_MODE_AUTO_POLLING_INTERVAL`	`integer`	If the `DISCOVERY_MODE` is set as `auto`, set the polling interval at which the Dashboard polls the discovery source for APIs. It is recommended to have a minimum value of 10 minutes.
`DISCOVERY_MODE_AUTO_DELETE_NON_DISCOVERED_APIS`	`string`	If the `DISCOVERY_MODE` is set as `auto`, you can choose to retain to delete APIs in ASE that are added manually. Set it to `true`, if you want to delete the APIs that are manually added in ASE.
`ASE_MODE`	`string`	Sets the mode in which ASE is deployed. Values can be either `inline` or `sideband`. Make sure this value is same as the value set in ASE.
`ABS_ACCESS_KEY`	`string`	The access key for the ABS admin user. For more information, see ABS users.
`ABS_SECRET_KEY`	`string`	The secret key for the ABS admin user. For more information, see ABS users.
`ABS_HOST`	`string`	The Internet Protocol (IP) address of ABS host.
`ENABLE_XPACK`	`string`	Configures whether X-Pack is installed. Default value is `true`. If the variable is set to `false`, the Web GUI protocol should be HTTP.
`ENABLE_SYSLOG`	`string`	Configures whether the Dashboard sends Syslog messages to the Syslog server. The default value is `false`. **Important** `ENABLE_SYSLOG` and `ENABLE_UI` both cannot be `false` at the same time. When `ENABLE_SYSLOG` environment variable is passed to the container, `SYSLOG_HOST`, and `SYSLOG_PORT` should also be passed. These are to configure the Syslog server and port number.
`ABS_RESTRICTED_USER_ACCESS`	`true/false`	Set to `true` if you want to use an ABS restricted user. For more information on restricted users, see ABS users.
`ABS_URL`	`string`	The URL should be in the form of https://<*IP*>:<*port*>. The URL is used by WebGUI to connect to ABS.
`ASE_URL`	`string`	The URL should be in the form of https://<*IP*>:<*port*>. The URL is used by WebGUI to connect to ASE.
`ASE_ACCESS_KEY`	`string`	The access key of the ASE admin user.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`ASE_SECRET_KEY`	`string`	The secret key of the ASE admin user.
`DASHBOARD_URL`	`string`	The URL should be in the form of https://*<IP>*:*<port>*. The URL is used by WebGUI to connect to the Dashboard. IP and port number are for Kibana.
`H2_DB_PASSWORD`	`string`	The password for the H2 database.
`H2_DB_ENCRYPTION_PASSWORD`	`string`	The password to change the encryption method of the H2 database.
`WEBGUI_ADMIN_PASSWORD`	`string`	The password for the admin user of WebGUI.
`WEBGUI_PING_USER_PASSWORD`	`string`	The password for `ping_user` of WebGUI.
`SESSION_MAX_AGE`	`6h`	Defines the maximum time for a session. The configured values should be in the form of *<number><duration_suffix>*. The duration should be > 0. Allowed `duration_suffix` values are `m` for minutes, `h` for hours, and `d` for days.
`MAX_ACTIVE_SESSIONS`	`50`	Defines the maximum number of active UI sessions at any given time. The value should be greater than 1.
`AUTHENTICATION_MODE`	`native` or `sso`	Set the value to `sso` to authenticate the Dashboard with PingFederate.
`SSO_OIDC_CLIENT_ID`	`string`	Client ID value configured in the identity provider.
`SSO_OIDC_CLIENT_SECRET`	`string`	Client secret configured for the corresponding Client ID.
`SSO_OIDC_CLIENT_AUTHENTICATION_METHOD`	`BASIC`, `POST`, and `NONE`	OpenID Connect (OIDC) Client authentication mode. The valid values are `BASIC`, `POST`, or `NONE`.
`SSO_OIDC_PROVIDER_ISSUER_URI`	`string`	The PingFederate URI that is required by WebGUI to establish single sign-on (SSO). The default value is `https://127.0.0.1:9031`. **Note** The PingIntelligence Dashboard Docker image can be generated by packaging it with the PingFederate public certificate. To do so, the certificate needs to be placed in `certs/webgui` directory with the name `webgui-sso-oidc-provider.crt`.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`SSO_OIDC_PROVIDER_USER_UNIQUEID_CLAIM_NAME`	`string`	Claim name for the unique ID of the user in the UserInfo response. A new user is provisioned using this unique ID value.
`SSO_OIDC_PROVIDER_USER_FIRST_NAME_CLAIM_NAME`	`string`	Claim name for the first name of the user in the UserInfo response. Either first name or last name can be empty, but both should not be empty.
`SSO_OIDC_PROVIDER_USER_LAST_NAME_CLAIM_NAME`	`string`	Claim name for the last name of the user in the UserInfo response. Either first name or last name can be empty, but both should not be empty.
`SSO_OIDC_PROVIDER_USER_ROLE_CLAIM_NAME`	`string`	Claim name for the role of the user in the UserInfo response. Valid values for roles are `ADMIN` and `REGULAR`.
`SSO_OIDC_PROVIDER_CLIENT_ADDITIONAL_SCOPES`	`string`	Additional scopes in the authorization request. Multiple scopes should be values separated with a comma (,). OpenID profile scopes are always requested.
`TIMEZONE`	`string`	Set the timezone of the Dashboard to either `local` or `UTC`. The default value is `utc`. **Note** Make sure `TIMEZONE` is set to the same value in ASE, ABS, and the Dashboard.
`KAFKA_SERVERS`	`string`	Kafka `ip:port` needs to be configured.
`DE_CONSUMER_USER`	`string`	The data engine consumer user in Kafka.
`DE_CONSUMER_PASSWORD`	`string`	Consumer user password.
`DE_CONSUMER_GROUP`	`string`	The group in Kafka for the data engine consumer.
`TRANSACTIONS_TOPIC`	`string`	ABS transaction topic in Kafka.
`ATTACK_TOPIC`	`string`	ABS attack topic in Kafka.
`ELASTIC_URL`	`string`	External Elasticsearch URL.
`ELASTIC_PASSWORD`	`string`	External Elasticsearch password.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`ELASTIC_USERNAME`	`string`	External Elasticsearch username.

API Publish Environment Variables

The following table lists the API Publish environment variables and the values.

Environment	Value	Usage
`MONGO_USERNAME`	`string`	MongoDB username.
`MONGO_PASSWORD`	`string`	MongoDB password.
`MONGO_CERTIFICATE`	`string`	Set to `true` if the MongoDB instance is configured in Secure Sockets Layer (SSL) mode, and you want to do the server certificate verification.
`MONGO_AUTH_MECHANISM`	`string`	MongoDB authentication: • Supported Mongo authentication mechanisms are: ◦ `DEFAULT`: Provide `MONGO_USERNAME` and `MONGO_PASSWORD`. ◦ `PLAIN`: Provide the external Lightweight Directory Access Protocol (LDAP) username and password in `MONGO_USERNAME` and `MONGO_PASSWORD`. • Set to `NONE` if authentication is not enabled in Mongo.
`MANAGEMENT_PORT`	`string`	Port for the API Publish service
`APIPUBLISH_JKS_PASSWORD`	`string`	The API Publish password for the JKS file. You can change the password, and it will be generated during installation.
`MONGO_SSL`	`string`	Indicates whether SSL is used for Mongo. The default value is `false`.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`DATABASE_NAME`	`string`	Database name.
`META_DATABASE`	`string`	Meta database name.
`APIPUBLISH_CLI_ADMIN_PASSWORD`	`string`	API Publish command-line interface (CLI) password.

Kafka Environment Variables

The following table lists the Kafka environment variables and the values.

Environment	Value	Usage
`ZOOKEEPER_URL`	`<IP or hostname>:port`	Zookeeper URL.
`KAFKA_SSL_PORT`	`string`	SSL port for Kafka.
`KAFKA_SASL_PORT`	`string`	SASL port for Kafka.
`KAFKA_MIN_INSYNC_REPLICA`	`string`	The minimum number of in-sync replicas for data in Kafka.
`ABS_CONSUMER_USER`	`string`	ABS consumer user in Kafka.
`ABS_PRODUCER_USER`	`string`	ABS producer user in Kafka.
`ABS_CONSUMER_PASSWORD`	`string`	ABS consumer user password.
`ABS_PRODUCER_PASSWORD`	`string`	ABS producer user password.
`ABS_CONSUMER_GROUP`	`string`	ABS group in Kafka.
`DE_CONSUMER_USER`	`string`	Data engine consumer user in Kafka.
`DE_CONSUMER_PASSWORD`	`string`	Consumer user password.
`DE_CONSUMER_GROUP`	`string`	Group in Kafka for the data engine consumer.
`RETENTION_PERIOD`	`string`	Retention period of data in topics.
`POD_NAME`	`string`	Kafka broker ID.

Zookeeper Environment Variables

The following table lists the Zookeeper environment variables and the values.

Copyright © 2025 Ping Identity Corporation

Environment	Value	Usage
`ZOOKEEPER_PORT`	`string`	Non-SSL port for Zookeeper.
`ZOOKEEPER_SSL_PORT`	`string`	Non-SSL port for Zookeeper.

Using environment variables example

The following sections show example of using environment variables to create containers.

About this task

The containers must be created in the following order:

1. MongoDB
2. Zookeeper
3. Kafka
4. API Behavioral Security (ABS)
5. API Security Enforcer (ASE)
6. API Publish
7. Dashboard

Steps

1. To launch the MongoDB container, run the following command with some sample environment variables:

   ```
   docker run -d --name mongo --hostname mongo
   ```

   Result:

   Running this command creates the MongoDB container with settings in environment variables provided.

   ```
   docker run -d --name mongo --hostname mongo -e MONGO_USERNAME="new_mongo_user" \
   -e MONGO_PASSWORD="new_mongo_password" \
   -e WIRED_TIGER_CACHE_SIZE_GB="1.8" \
   -e MONGO_SSL="true" pingidentity/mongo:4.2.0
   ```

   If any of the environment variables are not used, then the container is launched with the default values.

2. To launch the Zookeeper container, run the following command with some sample environment variables:

   ```
   docker run -d --name zookeeper --hostname zookeeper
   ```

   Example:

Copyright © 2025 Ping Identity Corporation

```
docker run -d --name zookeeper --hostname zookeeper -e ZOOKEEPER_PORT="2181" \
-e ZOOKEEPER_SSL_PORT="2182" \
pingidentity/zookeeper:5.1
```

3. To launch the Kafka container, run the following command with some sample environment variables:

```
docker run -d --name kafka --hostname kafka --link zookeeper:zookeeper
```

Example:

```
docker run -d --name kafka --hostname kafka --link zookeeper:zookeeper \
-e ZOOKEEPER_URL="zookeeper:2182" \
-e KAFKA_SASL_PORT="9093" \
-e KAFKA_SSL_PORT="9092" \
-e DELETE_TOPIC="true" \
-e REPLICATION_FACTOR="1" \
-e TOPIC_PARTITION="1" \
-e ABS_CONSUMER_USER="abs_consumer" \
-e ABS_PRODUCER_USER="abs_producer" \
-e ABS_CONSUMER_GROUP="pi4api.abs" \
-e ABS_CONSUMER_PASSWORD="changeme" \
-e ABS_PRODUCER_PASSWORD="changeme" \
-e TRANSACTION_TOPIC="pi4api.queuing.transactions" \
-e ATTACK_TOPIC="pi4api.queuing.ioas" \
-e ANOMALIES_TOPIC="pi4api.queuing.anomalies" \
-e DE_CONSUMER_USER="pi4api_de_user" \
-e DE_CONSUMER_GROUP="pi4api.data-engine" \
-e DE_CONSUMER_PASSWORD="changeme" \
-e RETENTION_PERIOD="172800000" \
-e POD_NAME="0" \
pingidentity/kafka:5.1
```

4. To launch the ABS container, run the following command with some sample environment variables:

```
docker run -d --name abs --hostname abs --link mongo:mongo --link kafka:kafka
```

Example:

Copyright © 2025 Ping Identity Corporation

```
docker run -d --name abs --hostname abs --link mongo:mongo --link kafka:kafka -e MONGO_RS=mongodb://mongo:
27017 \
-e MONGO_USERNAME="new_mongo_user" \
-e MONGO_PASSWORD="new_mongo_password" -e MONGO_SSL="true" \
-e ABS_ACCESS_KEY="new_abs_ak" \
-e ABS_SECRET_KEY="new_abs_sk" -e ABS_ACCESS_KEY_RU="new_abs_ak_ru" \
-e ABS_SECRET_KEY_RU="new_abs_sk_ru" \
-e ATTACK_INITIAL_TRAINING="24" \
-e API_DISCOVERY="true" -e API_DISCOVERY_INITIAL_PERIOD="6" \
-e API_DISCOVERY_UPDATE_INTERVAL="1" \
-e API_DISCOVERY_SUBPATH="3" \
-e KAFKA_SERVERS="kafka:9093" \
-e KAFKA_MIN_INSYNC_REPLICA="1" \
-e ABS_CONSUMER_USER="abs_consumer" \
-e ABS_PRODUCER_USER="abs_producer" \
-e ABS_CONSUMER_GROUP="pi4api.abs" \
-e ABS_CONSUMER_PASSWORD="changeme" \
-e ABS_PRODUCER_PASSWORD="changeme" \
-e TRANSACTION_TOPIC="pi4api.queuing.transactions" \
-e ATTACK_TOPIC="pi4api.queuing.ioas" \
-e ANOMALIES_TOPIC="pi4api.queuing.anomalies" \
pingidentity/abs:5.1
```

5. To launch the ASE container, run the following command with some sample environment variables to launch the ASE container:

```
docker run -d --name ase --link abs:abs --hostname ase
```

Example:

```
docker run -d --name ase --link abs:abs --hostname ase -e MODE="inline" \
-e ENABLE_CLUSTER="true" -e ENABLE_ABS="true" -e ABS_ENDPOINT="abs:8080" \
-e ABS_ACCESS_KEY="new_abs_ak" -e ABS_SECRET_KEY="new_abs_sk" -e ENABLE_ABS_PUBLISH="true" --shm-size=1g
pingidentity/ase:5.1
```

6. To launch the second ASE node in ASE cluster, run the following command with some sample environment variables to launch the ASE node in a cluster:

```
docker run -d --name ase1 --link abs:abs --link ase:ase --hostname ase1
```

Example:

```
docker run -d --name ase1 --link abs:abs --link ase:ase --hostname ase1 \
-e MODE="inline" -e ENABLE_CLUSTER="true" \
-e PEER_NODE="ase:8020" -e ENABLE_ABS="true" \
-e ABS_ENDPOINT="abs:8080" -e ABS_ACCESS_KEY="new_abs_ak" \
-e ABS_SECRET_KEY="new_abs_sk" --shm-size=1g pingidentity/ase:5.1
```

7. To launch the API Publish container, run the following command with some sample environment variables:

Copyright © 2025 Ping Identity Corporation

```
docker run -d --name apipublish --hostname apipublish --link mongo:mongo
```

Example:

```
docker run -d --name apipublish --hostname apipublish --link mongo:mongo -e MONGO_RS=mongodb://mongo:27017 \
-e MONGO_USERNAME="new_mongo_user" \
-e MONGO_PASSWORD="new_mongo_password" -e MONGO_SSL="true" \
pingidentity/apipublish:5.1
```

8. To launch the Dashboard, run the following command with some sample environment variables:

```
docker run -d --name webgui --link abs:abs --link ase:ase --link apipublish:apipublish --link kafka:kafka --
hostname webgui
```

Example:

```
docker run -d --name webgui --link abs:abs --link ase:ase --link apipublish:apipublish --link kafka:kafka --
hostname webgui \
-e ABS_RESTRICTED_USER_ACCESS="false" \
-e ABS_ACCESS_KEY="new_abs_ak" -e ABS_SECRET_KEY="new_abs_sk" -e ABS_HOST="abs" \
-e ABS_URL="https://abs:8080" -e ASE_URL="https://ase:8010" \
 -e WEBGUI_ADMIN_PASSWORD="new_webgui_admin_password" \
-e WEBGUI_PING_USER_PASSWORD="new_webgui_pinguser_password" \
-e KAFKA_SERVERS="kafka:9093" \
-e KAFKA_MIN_INSYNC_REPLICA="1" \
-e DE_CONSUMER_USER="ping_user" \
-e DE_CONSUMER_GROUP="pi4api.data-engine" \
-e DE_CONSUMER_PASSWORD="changeme" \
-e TRANSACTION_TOPIC="pi4api.queuing.transactions" \
-e ATTACK_TOPIC="pi4api.queuing.ioas" \
-e ANOMALIES_TOPIC="pi4api.queuing.anomalies" \
-e API_PUBLISH_URL="https://apipublish:8050" \
-p 8030:8030 -p 8040:8040 pingidentity/dashboard:5.1
```

Next steps

When the containers are created, the exposed ports are not mapped. To map the ports, you need to complete port mapping using the `-p` option in the **`docker run`** command. The following table lists the ports that should be exposed in the container.

Component	Port number	Usage
ASE	8080	HTTP data plane
	8443	HTTPS data plane
	8010	Management port number
	8020	Cluster port number
ABS	8080	API server port number

Copyright © 2025 Ping Identity Corporation

Component	Port number	Usage
Dashboard	8030	Dashboard port number
MongoDB	27017	MongoDB port number
API Publish	8050	API Publish port number
Zookeeper	2181 2182	Zookeeper port number
Kafka	9092 9093	Kafka port number

PingIntelligence Dashboard ILM policy configuration

The Index Lifecycle Management (ILM) policy allows for an automatic rollover of index based on time or size of data.

The `ilm.json` file contains the configurations of the policy. The following table lists the variables that you can set in the `ilm.json` file. For more information on `ilm.json` configuration, see Automatic rollover index.

Variable	Description
`max_size`	Defines the maximum size of the Elasticsearch rollover index. When the index size reaches the defined value, it roll overs. The `max_size` value should be a positive non-zero number. Allowed units are MB and GB.
`max_age`	Defines the maximum age of the Elasticsearch rollover index configuration. The `max_age` value should be a positive non-zero number. Allowed units are `h` for hours and `d` for the number of days. If both `max_size` and `max_age` are configured, then the index rolls over based on the value that is achieved first.
`min_age`	Defines the minimum age, after which the Elasticsearch rollover index enters into a different phase. Allowed units are `h` for hours and `d` for the number of days. Every index starts from the `hot` phase. For more information on the phases in an index life cycle, see Automatic rollover index.
`priority`	Defines the sequence in which indices are reloaded back into memory when Elasticsearch restarts. Use a positive integer number to set the priority.

The following is snippet of a sample `ilm.json` file:

Copyright © 2025 Ping Identity Corporation

```
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_size": "7GB",
            "max_age": "7d"
          },
          "set_priority": {
            "priority": 100
          }
        }
      },
      "warm": {
        "min_age": "30d",
        "actions": {
          "set_priority": {
            "priority": 50
          }
        }
      },
      "cold": {
        "min_age": "90d",
        "actions": {
          "freeze": {},
          "set_priority": {
            "priority": 0
          }
        }
      }
    }
  }
}
```

PingIntelligence Production Deployment

PingIntelligence automated deployment for virtual machines and servers

PingIntelligence software combines real-time application programming interface (API) security and artificial intelligence (AI) analytics to detect, report, and block cyberattacks on data and applications exposed through APIs.

The software consists of three platforms: API Security Enforcer (ASE), API Behavioral Security (ABS) AI Engine, and PingIntelligence Dashboard.

This section describes the installation and execution of an Ansible package that automatically builds a PingIntelligence for APIs environment. The package installs and configures the following components:

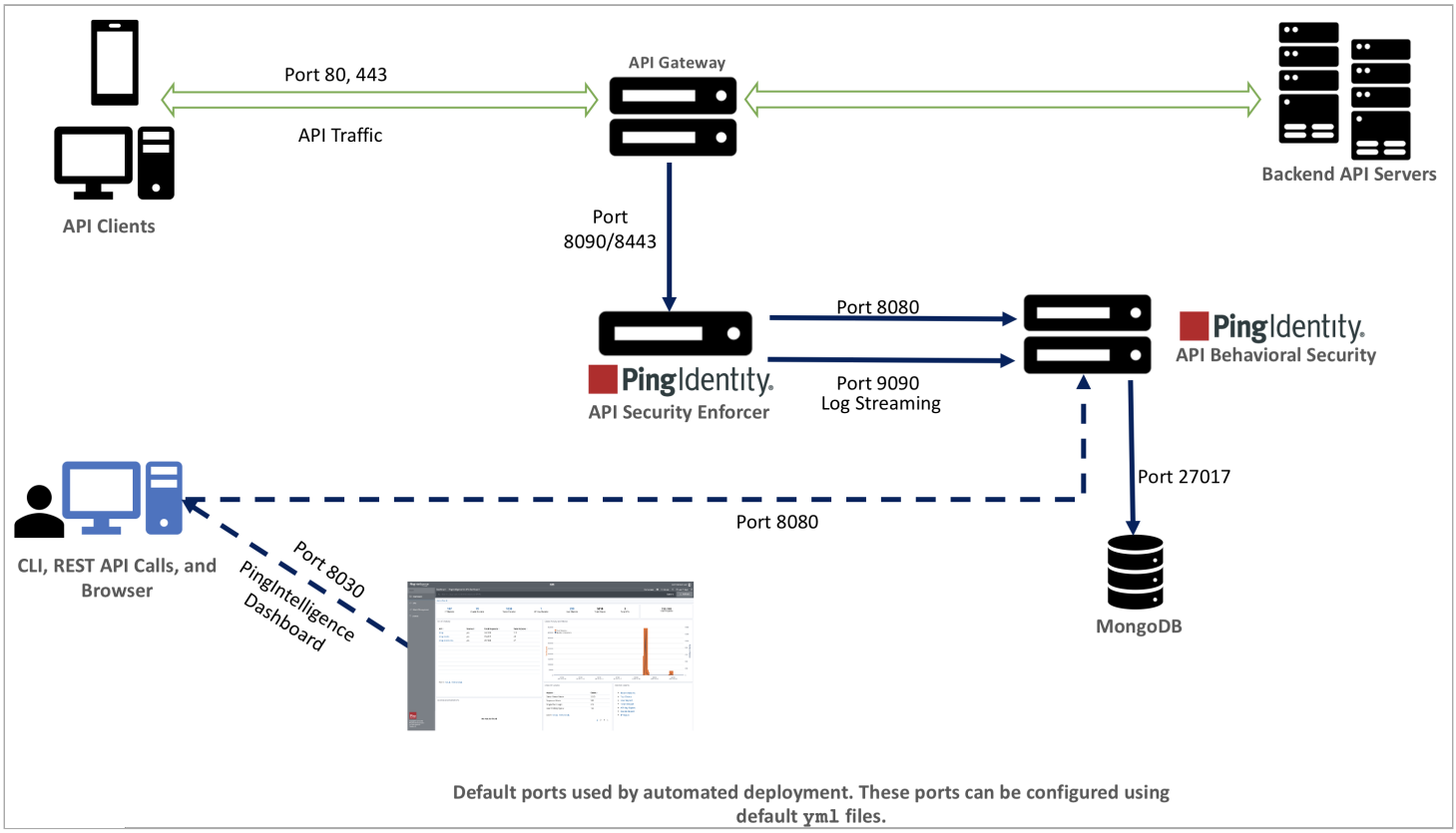
- ASE (inline or sideband configuration)
- ABS AI Engine
- MongoDB database

Copyright © 2025 Ping Identity Corporation

- PingIntelligence for APIs Dashboard

The supported operating systems are RHEL 7.9 and 8 or Ubuntu 18.04 LTS.

The following diagram shows an example of a sideband deployment.



API Security Enforcer (ASE)

Applies real-time API metadata ingestion and enforces optional blocking. ASE can be deployed in inline or sideband mode and works with the ABS engine to identify attacks. For more information, see Inline ASE and Sideband ASE.

API Behavioral Security AI engine

Executes AI algorithms to detect in near real-time cyberattacks targeting data, applications, and systems with APIs. Attack information can be automatically pushed to all ASEs to block ongoing breaches and prevent reconnection.

PingIntelligence for APIs Dashboard

Provides:

- Graphical view of an API environment, including user activity, attack information, and deny-listed clients, using Elasticsearch and Kibana
- Visibility into API activity
- API training status and other information
- API Discovery management using automatic or manual mode
- Attack insight to understand why a client was flagged for an attack

Copyright © 2025 Ping Identity Corporation

- Attack management through unblocking clients or tune AI Engine thresholds
- ABS license information

Administrators

You can install all of the PingIntelligence products either as a user with `sudo` access or a normal user without `sudo` access.

> **Important**
>
> Make sure that the entire deployment is a homogenous deployment. Install all of the products as either a `sudo` user or as a normal user, not a mix of both.

Time zone

All PingIntelligence components (ASE, ABS AI Engine, and Dashboard) should be installed using the same time zone, either local or UTC. MongoDB should also be configured to the same time zone as PingIntelligence components.

PingIntelligence deployment modes

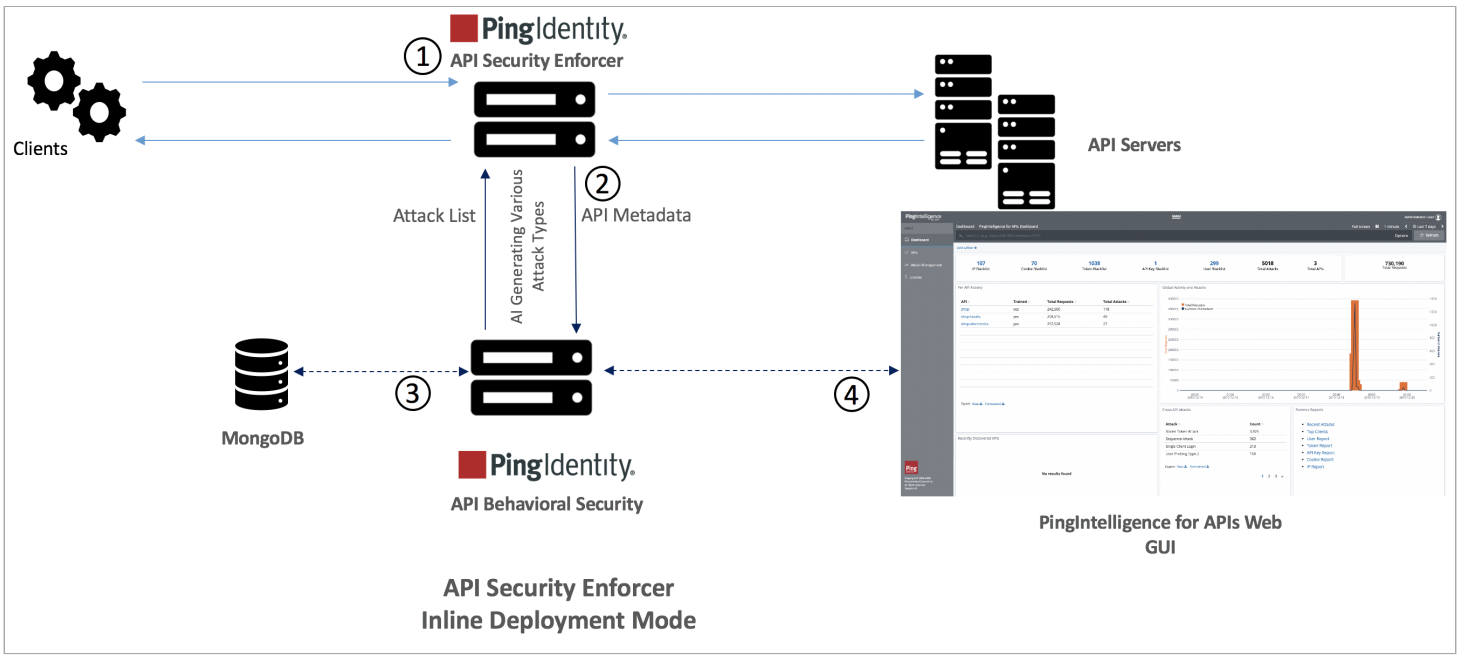
PingIntelligence supports inline and sideband deployment modes.

Inline mode

In PingIntelligence inline deployment mode, API Security Enforcer (ASE) sits at the edge of your network to receive the application programming interface (API) traffic. Inline mode can also be deployed behind an existing load balancer, such as AWS Elastic Load Balancing (ELB).

In inline mode, ASE deploys at the edge of the data center and terminates SSL connections from API clients. It then forwards the requests directly to the APIs, API gateways, or app servers, such as Node.js, WebLogic, Tomcat, and PHP.

To configure ASE to work in inline mode, set the `mode=inline` in the `ase-defaults.yml` file.



Copyright © 2025 Ping Identity Corporation

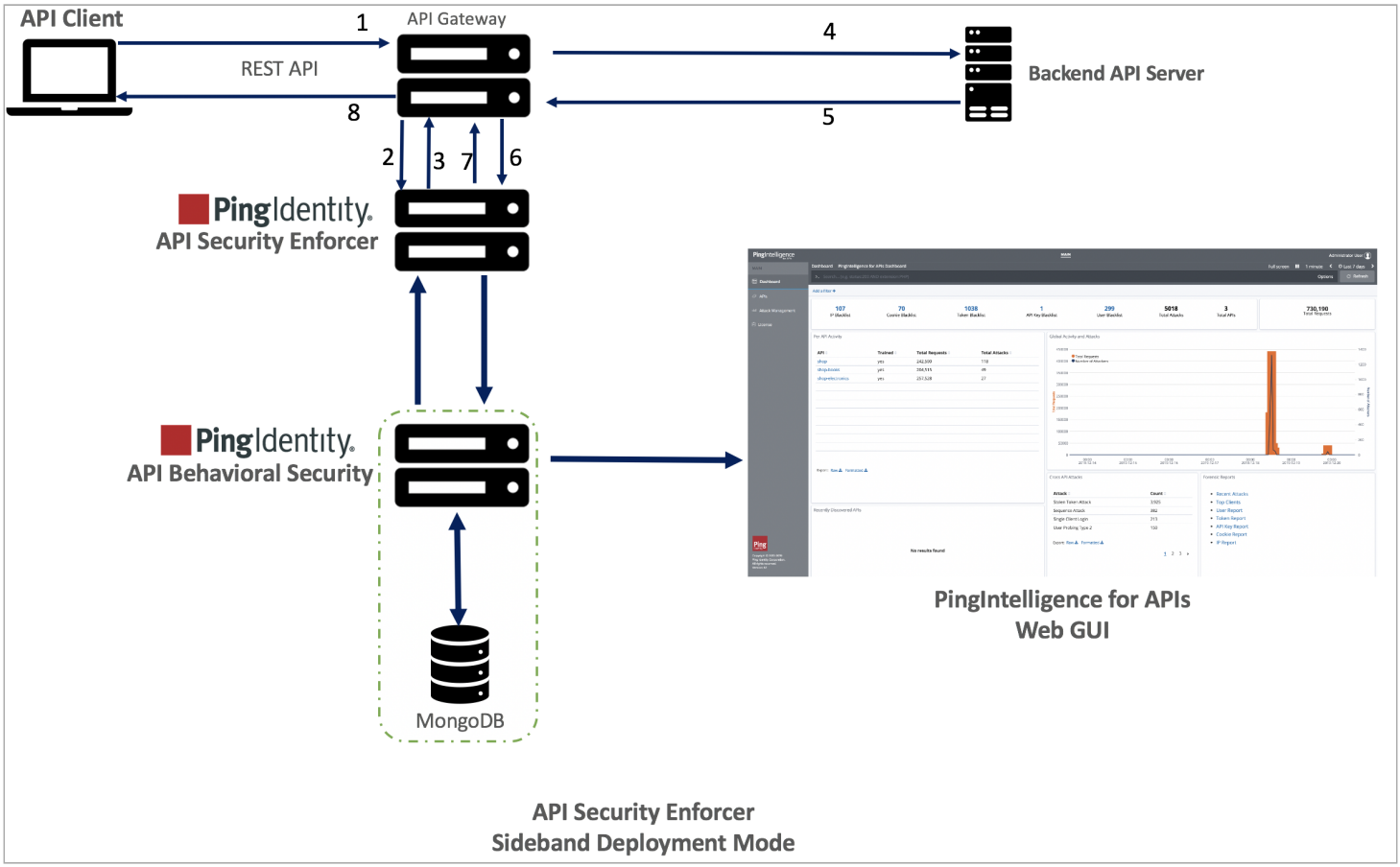
The following is a high-level description of traffic flow:

1. A client request is received by ASE. The request is logged in the access log file. ASE then forwards the request to the backend server. The response is received by ASE and logged in the access log file.
2. The request and response in the access log file are sent to the ABS artificial intelligence (AI) engine for processing. The ABS AI engine generates the attack list, which is fetched by ASE. The future requests received by ASE are either forwarded to the backend server or blocked by ASE based on the attack list.
3. The AI engine data is stored in MongoDB.
4. The PingIntelligence for APIs WebGUI fetches the data from ABS to display in the dashboard.

Sideband mode

When PingIntelligence is deployed in sideband mode, a sideband policy is added to the API gateway, which makes calls to ASE to pass API request and response metadata. In sideband mode, ASE does not terminate the client requests.

To configure ASE to work in sideband mode, set the `mode=sideband` in the `ase-defaults.yml` file.



The following is a description of the traffic flow through the API gateway and Ping Identity ASE:

1. The API client sends a request to the API gateway.
2. The API gateway makes an API call to send the request metadata in JavaScript Object Notation (JSON) format to ASE.

Copyright © 2025 Ping Identity Corporation

3. ASE checks the request against a registered set of APIs and checks the client identifier against the AI-generated deny list. If all checks pass, ASE returns a `200-OK` response to the API gateway. Otherwise, a different response code is sent to the gateway. The request is also logged by ASE and sent to the AI engine for processing.
4. When the API gateway receives a response from ASE, then it forwards the request to the backend server unless blocking is enabled and the client is on the deny list.
5. The response from the backend server is received by the API gateway.
6. The API gateway makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` response to the API gateway.
8. The API gateway sends the response received from the backend server to the client.

> **Note**
>
> To complete the ASE sideband mode deployment, see Integrate API gateways for sideband deployment.

PingIntelligence automated deployment preparation

Before you can run the PingIntelligence for APIs automated deployment script, you must familiarize yourself with the requirements.

Commonly used terms for deployment machines

Management machine

Management host machine or a management machine is the server on which the PingIntelligence for APIs automated deployment script is downloaded and run.

Host machine

Server or servers where PingIntelligence for APIs components are installed.

Prerequisites

Management machine operating system

PingIntelligence for APIs automated deployment requires installing the following on the management machine:

- RHEL 7.9
- PIP3

Host machine operating system

PingIntelligence for APIs host machine operating system can be RHEL 7.9 and 8 or Ubuntu 18.04 LTS.

Copyright © 2025 Ping Identity Corporation

Important

For all the provisioned host machines, make sure the deployment is homogenous with respect to the operating systems and their versions. For example, if the host machines are provisioned to run RHEL 7.9, all host machines must be run RHEL 7.9. Do not create a setup with a mixture of deployments across host machines.

Ansible

The management host machine must have Ansible 2.10.0 or later installed.

Note

Ansible 2.10 and later distributes two artifacts:

- A community package called `ansible`
- A minimalist language and runtime called `ansible-core` (called `ansible-base` in version 2.10).

As some required packages are not available in `ansible-base`, the `ansible` community package is also required. When installing the `ansible` community version 2.10.0, running the `ansible --version` command returns the following output:

```
ansible --version
ansible 2.10.17
 config file = None
 configured module search path = ['/home/abhalla/.ansible/plugins/modules', '/usr/share/ansible/
plugins/modules']
 ansible python module location = /usr/local/lib/python3.6/site-packages/ansible
 executable location = /usr/local/bin/ansible
 python version = 3.6.8 (default, Sep 26 2019, 11:57:09) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]
```

The `ansible` 2.10.0 community package includes `ansible-base` 2.10.17.

Python

The management host machine must have Python installed. The supported version is Python 3.8.13.

Install Python as the root user, and make sure Python 3 points to Python 3.8.13 (for example, by creating a soft link with `ln -s /usr/local/bin/python3.8 /usr/bin/python3`).

Machine learning (ML) service node

Install Python 3.8.13 and PIP3 on the ML service node. Make sure PIP3 points to Python 3.8.13.

User

Automated installation requires a user with passwordless authentication for Secure Shell (SSH) connection to the host machines. User should also have passwordless `sudo` access to all the host machines. Alternatively, you can also set up a user with password by editing the `hosts` file. For more information on the `hosts` file, see Configuring the `hosts` file and downloading software.

Copyright © 2025 Ping Identity Corporation

`firewalld` *package*

All the host machines should have an active `firewalld [python 3.8.13]` package on both Ubuntu and RHEL machines. If the package is not available, then manually open the ports that are used in the deployment. For more information on ports, see Change default settings.

Ubuntu machines

If you are deploying the setup on a Ubuntu machine, make sure that the MongoDB host machine has `libcurl4-openssl-dev` .

No pre-existing Java installations

Make sure that there are no pre-existing Java installations on the host machines. You can use the `# java -version` command to verify this.

> **Important**
>
> Uninstall all existing versions of Java from the host machines before proceeding with the installation of PingIntelligence for APIs components.

`libselinux-python`

Each node (management and host) must have the `libselinux-python` package installed.

Downloading the deployment package

The automated deployment installs different PingIntelligence components from this management machine.

Before you begin

PingIntelligence automated deployment requires a RHEL 7.9 management host machine to start the deployment.

About this task

To set up the management machine:

Steps

1. Sign on to the management machine as a root user.
2. Download the Ansible deployment package and save it to the `/opt` directory.
3. Untar the downloaded file:

   ```
   #tar -xf /opt/pi-api-deployment-<version>.tar.gz
   ```

   Result:

   Untarring the file creates the following subdirectories in the `pi-api-deployment` directory.

Copyright © 2025 Ping Identity Corporation

Directory	Description
`ansible`	Contains the different `.yml` files.
`bin`	Contains the `start.sh` and `stop.sh` scripts. Do not edit the contents of this directory.
`certs`	Contains API Security Enforcer (ASE), API Behavioral Security (ABS), Elasticsearch, Kibana, Dashboard, and MongoDB self-signed certificates and keys. Elasticsearch and Kibana certificates are in the Dashboard directory. **Note** If you want to use your own certificates and keys, replace the default certificates and keys with your certificates. Use the same file names as that of the files present in the `certs` directory. Make sure that the keys are passwordless.
`config`	Contains the default settings file for ASE, ABS, and Dashboard. These files are used to configure the various variables for installing PingIntelligence components.
`data`	System directory. Do not edit the contents of this directory.
`util`	Contains utilities to run PingIntelligence components as a service.
`external`	The third-party components, such as MongoDB, are downloaded in the `external` directory.
`keys`	After the installation is complete, the master keys of all the components are saved here.
`license`	Contains the PingIntelligence for APIs license file.
`logs`	Contains the log files for automated installation.
`software`	Contains the binary files for PingIntelligence components: ◦ ASE ◦ ABS ◦ Dashboard The directory also contains the `updated_packages` subdirectory that stores the PingIntelligence for APIs updated binaries with new master keys. You can use these binaries for future use.

Creating a new SSH user and configuring user authentication

Before you can connect to the host machines, you must configure user authentication. Optionally, you can create a new user.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> If you don't want to create a new user, you can use the default user configured in the `hosts` file.

When you configure user authentication, you can either configure passwordless authentication for the SSH user or use a password to connect to the host machines.

> **Note**
>
> The `sshpass` module must be installed on the RHEL host machine if you're authenticating using a password.

Creating a new user

About this task

If you do not have a user as mentioned in the PingIntelligence automated deployment preparation section, complete the following steps on all the provisioned host machines.

If you already have a user as described in Creating a new SSH user and configuring user authentication, start with Copying the PingIntelligence license.

Steps

1. Create an `ec2-user`.

   The `hosts` file in the automation package has `ec2-user` as the default user.

   1. **Optional:** To create your own username, run the `#useradd ec2-user` command.

2. Change the password by running the `#passwd ec2-user` command.

   > **Note**
   >
   > If you're installing PingIntelligence software as a non-sudo user, skip steps 3-5.

3. Add the user to the wheel group by running the `#usermod -aG wheel ec2-user` command.

4. Configure passwordless `sudo` access:

   ```
   #visudo
   %wheel ALL=(ALL) NOPASSWD: ALL
   ```

5. Verify the `/etc/ssh/sshd_config` file for `PubKeyAuthentication`.

   Troubleshooting:

   If it is set to `no`, then set it to `yes` and restart `sshd` service by the `#systemctl restart sshd` command.

Copyright © 2025 Ping Identity Corporation

Passwordless authentication

Setting up passwordless authentication
About this task

You can set up passwordless authentication from the management machine to other machines where PingIntelligence components are installed.

Steps

1. On the management machine, run the following command.

   > **Note**
   > The management machine is the machine from which the automated deployment script is run to deploy the various PingIntelligence software.

   ```
   # ssh-keygen -t rsa
   ```

   Result:

   This command generates the `ssh-keys`.

2. Accept all the default options. Make sure that you do not set the password for the key.

3. You have two options for configuring passwordless authentication:

   Choose from:

   - Run the **`ssh-copy-id`** command for each host machine but not the management machine:

     ```
     # ssh-copy-id pi-user@<ping-machine IPv4 address>
     ```

     For example (ping-ase):

     ```
     ssh-copy-id pi-user@192.168.11.148
     ```

   - Copy and add the `ssh-keys` manually:

     1. Fetch the generated key in step 1 from `/home/$USER/.ssh/id_rsa.pub`.
     2. Copy the key and add it to the `/home/$USER/.ssh/authrorized_keys` file on all the host machines where PingIntelligence components are installed.

        > **Important**
        > If configuring passwordless authentication does not succeed, contact your system administrator.

Copyright © 2025 Ping Identity Corporation

Password-based authentication

Setting up authentication using a password
Before you begin

Ensure that:

- You've installed `sshpass` module on the management host machine.

> **Note**
> The management host machine is a RHEL 7.6 machine.

- The password that you configure for the user in the `hosts` file must already be configured on the host machines.

About this task

You can also use password to authenticate with PingIntelligence and MongoDB host machines.

Configure the password of the host machine in the `hosts` file.

Steps

- To add the password in the `hosts` file, edit the `hosts` file to configure the password in `ansible_ssh_pass` parameter as shown in bold in the following `hosts` file snippet

```
# Ansible SSH user to access host machines
ansible_ssh_user=ec2-user
# Uncomment the ansible_ssh_pass line and configure password of ansible_ssh_user if you want
to use SSH connection with password.
# If you do not use this option, then the SSH user uses password-less authentication.
#ansible_ssh_pass=<SSH_user_password>
```

Verifying SSH connectivity

Steps

- Manually verify SSH connectivity between the management machine and the host machine by running the `ssh user@remote-machine "ls"` command.

Next steps

To continue your configuration, see Copying the PingIntelligence license.

Copying the PingIntelligence license

PingIntelligence API Security Enforcer (ASE) and API Behavioral Security (ABS) require a valid license to start.

Steps

- Copy the license file, `PingIntelligence.lic`, to the `/license/` directory.

Copyright © 2025 Ping Identity Corporation

Example:

The following snippet shows the structure of the license file:

```
ID=
Organization=
Product=
Version=
IssueDate=
EnforcementType=
ExpirationDate=
MaxTransactionsPerMonth=
Tier=
```

> **Note**
>
> Make sure that the license file is named `PingIntelligence.lic` .

hosts file and downloading software">

Configuring the `hosts` file and downloading software

The `hosts` file contains the various parameters to be configured for installation of PingIntelligence components.

About this task

The configuration file has parameters configured to download third-party components. If the management machine does not have internet access, download the Third-party components manually.

> **Note**
>
> Make sure that the entire deployment is homogenous with respect to the provisioned machines. Install all PingIntelligence components either on an RHEL machine or on Ubuntu machines. Do not install on a mix of RHEL and Ubuntu.

To configure the hosts file:

Steps

1. Configure the following fields in the `config/hosts` file.

Copyright © 2025 Ping Identity Corporation

Variable	Description
IP addresses ◦ *[ase]* ◦ *[abs]* ◦ *[mongodb]* ◦ *[dashboard]* ◦ *[elasticsearch]* ◦ *[api_publishing_service]* ◦ *[kafka]* ◦ *[abs_reporting_node]* ◦ *[webgui]*	Configure the following Internet Protocol (IP) addresses: ***[ase]*** ASE IP address ***[abs]*** ABS IP address ***[mongodb]*** MongoDB IP address and port. Providing the port number is mandatory. ***[dashboard]*** Dashboard IP address ***[elasticsearch]*** Elasticsearch IP address ***[api_publishing_service]*** API publishing service IP address ***[kafka]*** Kafka IP address ***[abs_reporting_node]*** ABS reporting node IP address **Important** The IP address for *[abs]* and*[abs_reporting_node]* should be different. If you are installing all the components on a single host, leave the *[abs_reporting_node]* field blank. ***[webgui]*** Web GUI IP address. Web GUI and dashboard engine are part of the same package; however, you can install them on separate machines. If you want to install Web GUI and dashboard engine in the same machine, configure the same IP address in *[dashboard]* and *[webgui]*. **Note** If you are setting up an evaluation environment, all of the components (ASE, ABS, MongoDB, Dashboard, WebGUI, and ElasticSearch) can share a single IP address. Leave the `abs_reporting_node` field blank when all the components have the same IP address. For production deployments: ◦ ASE, ABS AI Engine, and MongoDB should be deployed on separate servers for redundancy. ◦ Dashboard, WebGUI, and ABS Reporting node (optional) can be deployed on a single server. ◦ Elasticsearch should be deployed on a standalone server.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`installation_path`	Configure the path where you want to install the PingIntelligence components to be. The default value is `/home/ec2-user`. **Important** The path that you provide in the `installation_path` variable must exist on the machine. The automation script does not create this path. If you are installing all the PingIntelligence components on different machines, manually create the same path on each machine before running the automation script.
`install_with_sudo`	When set to `false`, the script installs PingIntelligence for a normal user. When set to `true`, the script installs PingIntelligence as a root user if the port number of ports configured are less than 1024.
`install_as_service`	Set it to `true` if you want to install PingIntelligence components as a service. To install PingIntelligence components, you must be a `root` user. Set `install_with_sudo` as `true`. If you install PingIntelligence components as a service, the components are automatically restarted when the system is rebooted. Check the `ansible.log` file to verify starting PingIntelligence components as a service.
`install_mongo`	Set to `true` if you want the automated deployment to install MongoDB. Set it to `false` if you want to use an existing MongoDB installation. The default value is `true`. **Important** Configure the MongoDB IP address and port number even if `install_mongo` is set to `false`. MongoDB details are required to configure the `abs.properties` file.
`install_elasticsearch`	Set to `true` if you want the automated deployment to install Elasticsearch. Set to `false` if you want to use an existing Elasticsearch installation. The default value is `true`. **Note** ◦ If you set the option as `true`, provide an IP address in the `hosts` file for Elasticsearch. Leave the IP address blank in the `hosts` file if you configured the option as `false`. ◦ If you have configured the variable as `false`, configure the Uniform Resource Locator (URL) of your existing Elasticsearch in the `dashboard-defaults.yml` file. For more information, see Changing Dashboard default settings.

Copyright © 2025 Ping Identity Corporation

Variable	Description

`install_kafka`	◦ Automated deployment (default):
	■ Set `install_kafka = true` if you want the automated deployment to install Kafka.
	■ Provide an IP address in the `hosts` file for Kafka.
	◦ Existing Kafka installation:
	■ Set `install_kafka = false` if you want to use an existing Kafka installation.
	■ Leave the IP address blank in the `hosts` file.
	■ `kafka_server_url` : Configure the pre-existing Kafka IP port in the `config/abs-defaults.yml` file.
	`# When kafka is set to false in config/hosts, this url will be used`
	`# Give the host:port combination of mutiple kafka server in comma seperated.`
	`# Make sure kafka_server_url is accessible from ansible management host, dataengine, and abs nodes.`
	`#This will be used via dashboard dataengine module too.`
	`kafka_server_url: kafka_1:9093`
	■ `kafka_custom_truststore_password` :
	Set the `kafka_custom_truststore_password` parameter value in the `config/abs-defaults.yml` file with the password of your existing Kafka service. This must be set when `install_kafka` is set to `false` .
	`# When kafka is set to false in config/hosts, this password for jks will be used`
	`#This will be used via dashboard dataengine module too.`
	`kafka_custom_truststore_password: custom`
	◦ Place the existing Kafka `truststore.jks` file in the `cert_dir` directory.
	`cert_dir: "{{ root_dir }}/certs"`
	Note
	The default settings when you deploy Kafka and Zookeeper through the deployment framework are:
	◦ `ssl.hostnameVerification` is disabled
	◦ `allow.everyone.if.no.acl.found` is `true`

Copyright © 2025 Ping Identity Corporation

Variable	Description
`kafka_download_url`	Kafka download URL. A default URL is populated in the `hosts` file. **Note** If your machine does not have internet access, then download Kafka 2.12-2.5.0 and save the file as `kafka_2.12-2.5.0.tar.gz` in the `external` directory.
`jdk11_download_url`	The automated script requires OpenJDK 11.0.2. **Note** If your machine does not have internet access, download the OpenJDK 11.0.2 and save the file as `openjdk11.tar.gz` in the `external` directory.
`mongodb_download_url`	MongoDB download URL. A default URL is populated in the `hosts` file. **Note** 1. The default URL is RHEL version of MongoDB. If you are installing on Ubuntu, configure the MongoDB Ubuntu download URL. 2. If your machine does not have internet access, download the MongoDB 5.0.18 and save the file as `mongodb.tgz` in the `external` directory.
`elasticsearch_download_url`	Elasticsearch download URL. A default URL is populated in the `hosts` file. **Note** If your machine does not have internet access, download the Elasticsearch 7.13.4 and save the file as `elasticsearch-7.13.4.tar.gz` in the `external` directory.
`timezone`	Time zone setting for PingIntelligence components. It will set the time zone settings of ASE, ABS, and PingIntelligence for APIs Dashboard. Allowed values are `local` or `utc`. The default value is `utc`.
`ansible_ssh_user`	Ansible `ssh` user. The default value is `ec2-user`.
`ansible_ssh_pass`	Configure the Ansible SSH user's password if you want to use password to authenticate with the host machines. **Note** If you do not configure password, SSH use establishes a passwordless authenticated connection.

2. Add the Ansible username in the `ansible_ssh_user` field.

Copyright © 2025 Ping Identity Corporation

Example:

Copyright © 2025 Ping Identity Corporation

```
[ase]
10.96.6.41
10.96.6.111

[abs]
10.96.6.75
10.96.6.128

[api_publishing_service]
10.96.6.73

[abs_reporting_node]
10.96.6.73

[kafka]
10.96.6.63 zookeeper_id=1
10.96.6.160 zookeeper_id=2
10.96.6.254 zookeeper_id=3

[mongodb]
10.96.6.243 mongo_port=27017
10.96.6.236 mongo_port=27017
10.96.6.80 mongo_port=27017

[dataengine]
10.96.6.73

[elasticsearch]
10.96.6.10

[webgui]
10.96.6.73

[all:vars]

# Installation Path
installation_path="/home/ec2-user"

# install_as_service set to true will start ASE, ABS, Dashboard, Elasticsearch
# and kafka as systemd services.
install_as_service=true

# configure install_with_sudo to true if any of the ports used for ASE,
# ABS, Dashboard are < 1024. That component will be started using sudo.
# when install_as_service is true, install_with_sudo should be set to true.
install_with_sudo=true

# this option can be used if there is an existing mongo installation that can be used
# set it to false if Mongodb need not be installed
install_mongo=true

# this option can be used if there is an existing kafka installation that can be used
# set it to false if kafka need not be installed
install_kafka=true
```

Copyright © 2025 Ping Identity Corporation

```
# this option can be used if there is an existing elasticsearch installation that can be used.
# set it to false if elasticsearch need not be installed.
# when install_elasticsearch is set to false, remove any nodes under elasticsearch section and
# configure elasticsearch_url in config/dashboard-defaults.yml.
install_elasticsearch=true

# timezone setting. It will set timezone settings of ASE, ABS and Dashboard
# allowed values: local, utc
timezone=utc

# Download URLs for external packages
jdk11_download_url='https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_linux-
x64_bin.tar.gz'
mongodb_download_url='https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-5.0.18.tgz'
elasticsearch_download_url='https://artifacts.elastic.co/downloads/elasticsearch/
elasticsearch-7.13.4-linux-x86_64.tar.gz'
kafka_download_url='https://archive.apache.org/dist/kafka/2.5.0/kafka_2.12-2.5.0.tgz'

# Ansible SSH user to access host machines
ansible_ssh_user=ec2-user
# Uncomment the ansible_ssh_pass line and configure password of ansible_ssh_user if you wish to use
SSH connection with password.
# If you do not use this option, then the SSH user uses password-less authentication.
#ansible_ssh_pass=
```

The default value is `ec2-user`.

Manually downloading third-party components

If your management machine does not have internet access, you must manually download the third-party packages.

Before you begin

> **Important**
>
> If your management host machine has internet access, the automated deployment downloads the third-party components when you execute the deployment.

Steps

1. Install Ansible 2.10.17 or later on the management machine.

   > **Note**
   >
   > The management machine is the server from which the automated deployment script is run to deploy the various PingIntelligence components.

2. Install Python on the management host machine.
3. Download the following packages and copy them to the `external` directory using the specified names.

Copyright © 2025 Ping Identity Corporation

Package	Description
MongoDB	Download MongoDB 5.0.18 and save the file in the `external` directory as `mongodb.tgz`. ◦ MongoDB (RHEL) ◦ MongoDB (Ubuntu)
Elasticsearch	Download Elasticsearch from https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.3-linux-x86_64.tar.gz and save the file in the `external` directory as `elasticsearch-7.17.3.tar.gz`.
Kafka	Download Kafka from https://archive.apache.org/dist/kafka/3.5.2/kafka_2.13-3.5.2.tgz and save the file in the `external` directory as `kafka_2.13-3.5.2.tar.gz`.

Downloading the PingIntelligence for APIs software

Download the PingIntelligence software.

Steps

- Download the following PingIntelligence for APIs software to the `pi-api-deployment/software` directory:
  - API Security Enforcer (ASE) (RHEL or Ubuntu)
  - API Behavioral Security (ABS) AI Engine
  - PingIntelligence Dashboard

> **Note**
>
> ■ Do not change the name of the downloaded files.
> ■ The `software` directory should include the following files:
>
> ```
> -rw-r--r--. 1 pingidentity pingidentity 2.5M Jun 07 00:01 pi-api-dashboard-<version>.tar.gz
> -rw-r--r--. 1 pingidentity pingidentity 159M Jun 07 00:01 pi-api-abs-<version>.tar.gz
> -rw-r--r--. 1 pingidentity pingidentity 38M  Jun 07 00:01 pi-api-ase-rhel-<version>.tar.gz
> ```

Checking SSH connectivity

The Secure Shell (SSH) connectivity check provides details regarding the configured `user`, the IP address of the hosts for which SSH connectivity works or fails. Run the check before deploying PingIntelligence components.

Copyright © 2025 Ping Identity Corporation

About this task

To check the SSH connectivity from the management machine to other host machines:

Steps

- Enter the `$ ./bin/start.sh check` command on the management host command line.

```
User configured for SSH: ec2-user
Checking sudo connectivity between ansible management host and other hosts...
172.16.40.187 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
SSH connectivity to all hosts is successful
Capturing host information...
Host information is captured successfully
```

Troubleshooting

During SSH connectivity check between management host machine and PingIntelligence hosts, you might encounter errors because of user permission issues or connectivity issues between machines. The following are some of the common error messages:

```
User configured for SSH: ec2-user
Checking connectivity between ansible management host and other hosts...
172.16.40.187 | UNREACHABLE! => {
    "changed": false,
    "msg": "Authentication failure.",
    "unreachable": true
}
Sun Jul 12 19:22:41 MDT 2020: SSH connection error: connectivity to all hosts is not successful for ec2-
user
```

You have configured the user to use a password to authenticate with the hosts machines; however, the configured password in the `hosts` file is wrong.

```
User configured for SSH: ec2-user
Checking connectivity between ansible management host and other hosts...
172.16.40.187 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (publickey,password).\r\n",
    "unreachable": true
}
Sun Jul 12 19:26:16 MDT 2020: SSH connection error: connectivity to all hosts is not successful for ec2-
user
```

`ansible_ssh_pass` for authentication with password is uncommented in the `hosts` file. However, the password field has been left empty. Leaving the value empty is equivalent to passwordless authentication.

Copyright © 2025 Ping Identity Corporation

```
User configured for SSH: ec2-user
Checking sudo connectivity between ansible management host and other hosts...
172.16.40.187 | FAILED! => {
    "changed": false,
    "module_stderr": "Connection to 172.16.40.187 closed.\r\n",
    "module_stdout": "sudo: a password is required\r\n",
    "msg": "MODULE FAILURE",
    "rc": 1
}
Sun Jul 12 19:30:26 MDT 2020: SSH connection error: sudo connectivity to all hosts is not successful for
ec2-user
```

`install_with_sudo` is set to `true`, and there is an error connecting toPingIntelligence host machines.

Some probable reasons for error in connectivity are:

- The user is not in the `sudoers` file, or the user is not in any group that has `sudo` privileges.
- The user does not have `NOPASSWD: ALL` privileges in the `sudoers` file.

  ```
  User configured for SSH: ec2-user
  Checking sudo connectivity between ansible management host and other hosts...
  172.16.40.81 | UNREACHABLE! => {
      "changed": false,
      "msg": "Failed to connect to the host via ssh: ssh: connect to host 172.16.40.81 port 22:
  Connection timed out\r\n",
      "unreachable": true
  }
  Sun Jul 12 21:41:08 MDT 2020: SSH connection error: sudo connectivity to all hosts is not successful
  for ec2-user
  ```

  The IP address configured in the `hosts` file is not available.

  ```
  [localhost]: FAILED! => {"changed": false, "msg": "Aborting, target uses selinux but python bindings
  (libselinux-python) aren't installed!"}
    to retry, use: --limit @/home/ec2-user/411/pingidentity/pi-api-deployment/ansible/setup.retry
  ```

  `selinux` dependency: If you encounter the following error, you must install the `selinux` package on the host machine on which you see this error. Check the machine mentioned before `FAILED!` in the output to identify the machine where `selinux` needs to be installed.

Changing default settings

The deployment package provides `.yml` files to change the default settings of API Security Enforcer (ASE), API Behavioral Security (ABS), and the Dashboard.

Change the default settings before you execute the deployment package. For more information on each component, see API Security Enforcer, ABS AI Engine, and PingIntelligence Dashboard.

Copyright © 2025 Ping Identity Corporation

The following topics describe the default settings of each component:

- Changing ASE default settings
- Changing ABS default settings
- Changing Dashboard default settings
- Changing Kafka and Zookeeper default settings

> **Important**
>
> Make sure that the format of the default settings file is `.yml`.

Changing ASE default settings

You can change the default settings in ASE by editing the `ase-defaults.yml` file.

The following table lists the variables that you can set for ASE.

Variable	Description
`mode`	Sets the mode in which ASE is deployed. The default value is `inline`. Set the value to `sideband` if you want ASE to work in sideband mode.
`http_ws_port`	Data port used for HTTP or WebSocket protocol. The default value is 8000.
`https_wss_port`	Data port used for HTTPS or secure WebSocket protocol. The default value is 8443.
`management_port`	Management port used for command-line interface (CLI) and REST application programming interface (API) management. The default value is 8010.
`cluster_manager_port`	ASE node uses this port number to communicate with other ASE nodes in the cluster. The default value is 8020.
`keystore_password`	The password for ASE keystore. The default password is `asekeystore`.
`cluster_secret_key`	This key is used for authentication among ASE cluster node. All the nodes of the cluster must have the same `cluster_secret_key`. This key must be entered manually on each node of the ASE cluster for the nodes to communicate with each other. The default value is `yourclusterkey`.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`enable_ase_detected_attack`	This key is used to enable ASE to block auto detected attacks. Set this value to `true` to allow ASE to block auto detected attacks. The default value is `false`.
`enable_abs_attack`	This key is used to enable ASE to fetch attack list from ABS. Set this value to `true` to fetch the list from ABS. The default value is `false`.
`enable_sideband_keepalive`	This key is used only in ASE sideband mode. If set to `true`, ASE sends a keep-alive in response header for the TCP connection between API gateway and ASE. With the default `false` value, ASE sends a connection close in response header for connection between API gateway and ASE.
`Email default settings`	Configure the following settings: • `enable_emails`: Set it to `true` for ASE to send email notifications. Default value is false. • `smtp_host` and `smtp_port` • `sender_email`: Email address used from which email alerts and reports are sent. • `email_password`: Password of sender's email account. • `receiver_email`: Email address to which the email alerts and reports are sent.
`CLI admin password`	The default value for CLI admin is `admin`. To change the password, you need the current password.
`enable_abs_publish`	Determines whether the ASE fetches the published API list from ABS. Default: `true`
`abs_publish_request_minutes`	Determines in minutes how often ASE will get the published API list from ABS. Default: `10`
`enable_strict_request_parser`	Determines whether ASE parsing blocks requests with invalid header starts. Default: `true`

Important

Make a backup of the `ase-defaults.yml` file on a secure machine after the automated installation is complete.

The following is a sample `ase-defaults.yml` file.

Copyright © 2025 Ping Identity Corporation

```
---
ase:
 # Deployment mode for ASE. Valid values are inline or sideband
 mode: inline

 # Define ports for the PingIntelligence API Security Enforcer
 # Make sure ports are not same for single server installation
 http_ws_port: 8000
 https_wss_port: 8443
 management_port: 8010
 cluster_manager_port: 8020

 # Password for ASE keystore
 keystore_password: asekeystore

 # cluster_secret_key for ASE cluster
 cluster_secret_key: yourclusterkey

 # Set this value to true, to allow API Security Enforcer to block auto detected attacks.
 enable_ase_detected_attack: false
 # Set this value to true, to allow API Security Enforcer to fetch attack list from ABS.
 enable_abs_attack: true

 # enable keepalive for ASE in sideband mode
 enable_sideband_keepalive: false

 # Set this value to true, to allow API Security Enforcer to fetch published API list from ABS.
 enable_abs_publish: true

 #This value determines how often API Security Enforcer will get published API list from ABS.
 abs_publish_request_minutes: 10


 # enable strict parsing checks for client requests
 # If enabled, ASE will block request with invalid header start
 # If disabled, it will allow requests
 enable_strict_request_parser: true

 # Configure Email Alert. Set enable_emails to true to configure
 # email settings for ASE
 enable_emails: false
 smtp_host: smtp.example.com
 smtp_port: 587
 sender_email: sender@example.com
 email_password: password
 receiver_email: receiver@example.com

 # CLI admin password
 current_admin_password: admin
 new_admin_password: admin
```


Copyright © 2025 Ping Identity Corporation

Changing ABS default settings

You can change the default settings in ABS by editing the `abs-defaults.yml` file.

> **Important**
>
> Make a backup of the `abs-defaults.yml` file on a secure machine after the automated installation is complete.

The following is a sample `abs-defaults.yml` file.

Copyright © 2025 Ping Identity Corporation

```
---
abs:
 # Define ports for the PingIntelligence ABS
 # Make sure ports are not same for single server installation
 management_port: 8080

 # Mongo DB User and password
 mongo_username: absuser
 mongo_password: abs123
 # Define cache size for MongoDB (% of total RAM).
 # MongoDB will be configured to use this percentage of host memory.
 mongo_cache_size: 25
 # Communication between mongo and ABS
 mongo_ssl: true
 # Mongo DB Server Certificate Verification
 # Set to true if Mongo DB instance is configured in SSL mode and you want to do the server certificate
verification
 # By default ABS will not verify the MongoDB server certificate
 mongo_certificate_verify: false
 # Mongo replica set name
 mongo_replica_set: absrs01

 # When kafka is set to false in config/hosts, this url will be used
 # Give the host:port combination of mutiple kafka server in comma seperated.
 # Make sure kafka_server_url is accessible from ansible management host, dataengine, and abs nodes.
 #This will be used via dashboard dataengine module too.

 kafka_server_url: kafka_1:9093

 # When kafka is set to false in config/hosts, this passoword for jks will be used
 #This will be used via dashboard dataengine module too.

 kafka_custom_truststore_password: custom



 # Duration of initial training period (units in hours)
 # This value will be set in the mongo nodes
 attack_initial_training: 24

 # Memory for webserver and streaming server (unit is in MB)
 system_memory: 4096

 # Access keys and secret keys to access ABS
 access_key: abs_ak
 secret_key: abs_sk
 access_key_ru: abs_ak_ru
 secret_key_ru: abs_sk_ru

 # Password for ABS keystore
 jks_password: abs123

 #Users in Kafka for abs
 consumer_user: abs_consumer
```


Copyright © 2025 Ping Identity Corporation

```
  producer_user: abs_producer
  abs_groupid: pi4api.abs

  # Kafka Consumer Producer Password
  consumer_authentication_password: changeme
  producer_authentication_password: changeme

  #Kafka Relicas
  min_insync_replicas: 1
  #topics to be created in kafka
  transactions_topic: pi4api.queuing.transactions
  attacks_topic: pi4api.queuing.ioas
  anomalies_topic: pi4api.queuing.anomalies
  discovery_topic: pi4api.queuing.apis

   #Topic partition ,replication_factor and retention_period(in milli seconds)
   #These will be used when install_kafka is true and topics are created as part of deployment.
  topic_partitions: 1
  replication_factor: 1
  retention_period: 172800000

  # Configure Email Alert. Set enable_emails to true to configure
  # email settings for ABS
  enable_emails: false
  smtp_host: smtp.example.com
  smtp_port: 587
  sender_email: sender@example.com
  email_password: password
  receiver_email: receiver@example.com

  # CLI admin password
  current_admin_password: admin
  new_admin_password: admin

  poc_mode: false

 api_publishing_service:
  # Define ports for the PingIntelligence API Publish Service
  # Make sure ports are not same for single server installation
  management_port: 8050

  # Password for APIPublish keystore
  jks_password: api123

  # Mongo DB Server Certificate Verification
  # Set to true if Mongo DB instance is configured in SSL mode and you want to do the server certificate
 verification
  # By default apipublish will not verify the MongoDB server certificate
  mongo_certificate_verify: false

  server_ssl_key_alias: pingidentity

  # MongoDB Database names
  data_dbname: abs_data
  meta_database: abs_metadata
```

Copyright © 2025 Ping Identity Corporation

```
# MongoDB authentication
# If authentication is not enabled in MongoDB, set the mongo_auth_mechanism to NONE
# The supported MongoDB authentication mechanisms are DEFAULT and PLAIN.
# If authentication mechanism is DEFAULT, provide MongoDB username and password for mongo_username
# and mongo_password. If authentication mechanism is PLAIN, provide external
# LDAP username and password in mongo_username and mongo_password.
mongo_authentication_mechanism: DEFAULT

# CLI admin password
current_admin_password: admin
new_admin_password: admin
```

ABS variable settings

The following table lists the variables that you can set for ABS.

Variable	Description
`management_port`	Port for ABS to ASE and REST API to ABS communication. The default value is 8080.
`mongo_username` and `mongo_password`	MongoDB username and password. The default username is `absuser`, and the default password is `abs123`.
`mongo_cache_size`	If you are running all the PingIntelligence components on the same instance, keep the MongoDB cache size to a maximum of 25% of the system memory. If you are running MongoDB on a separate instance, keep the MongoDB cache size to a maximum of 40% of the system memory.
`mongo_ssl`	Default value is `true`. PingIntelligence deployment ships with a default self-signed certificate. Setting it to `false` establishes non-SSL connection between ABS and Mongo.
`mongo_certificate_verify`	Set it to `true` if you want to verify MongoDB Secure Sockets Layer (SSL) server certificate when ABS connects to MongoDB. The default value is `false`. **Note** Make sure `mongo_ssl` is set to `true` before setting `mongo_certificate_verify` to `true`.
`mongo_replica_set`	Name of the MongoDB replica set. Default name is `absrs01`.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`attack_initial_training`	The number of hours that you want to train the AI model before it moves to the prediction mode. Default value is 24 hours.
`system_memory`	Memory size in MB allocated to run machine learning jobs. Recommended to be at least 50% of system memory.
`access_key` and `secret_key`	The access key and secret for the admin user. For more information on different ABS users, see ABS users. **Note** ":" (colon) is a restricted character and not allowed in access key and secret key.
`access_key_ru` and `secret_key_ru`	The access key and secret for the restricted user. For more information on different ABS users, see ABS users. **Note** ":" (colon) is a restricted character and not allowed in access key and secret key.
`jks_password`	The password of the Java KeyStore (JKS). The default password is `abs123`.
`Email default settings`	Configure the following settings: • `enable_emails`: Set it to `true` for ASE to send email notifications. Default value is false. • `smtp_host` and `smtp_port` • `sender_email`: Email address used from which email alerts and reports are sent. • `email_password`: Password of sender's email account. • `receiver_email`: Email address at which the email alerts and reports are sent.
`CLI admin password`	The default value for command-line interface (CLI) admin is `admin`. To change the password, you need the current password.
`poc_mode`	Sets the mode in which the artificial intelligence (AI) engine sets the thresholds for the AI models. If set to `true`, AI engine sets thresholds at a lower value. It should be set to `true` only for a proof-of-concept deployment.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`consumer_user`	ABS consumer user in Kafka. Default: `abs_consumer`
`producer_user`	ABS producer user in Kafka. Default: `abs_producer`
`abs_groupid`	ABS group in Kafka. Default: `pi4api.abs`
`consumer_authentication_password`	ABS consumer user password. Default: `changeme`
`producer_authentication_password`	ABS producer user password. Default: `changeme`
`min_insync_replicas`	Minimum number of insync replicas for data in Kafka.
`transactions_topic`	ABS transaction topic in Kafka.
`attacks_topic`	ABS attack topic in Kafka.
`anomalies_topic`	ABS anomalies topic in Kafka.
`discovery_topic`	ABS discovery topic in Kafka.
`topic_partitions`	Number of partitions for topics.
`replication_factor`	Replication factor for topics.
`retention_period`	Retention period of data on topics.
`kafka_server_url`	Pre-existing Kafka `ip:port` that must be configured in `config/abs-defaults.yml`.
`kafka_custom_truststore_password`	Pre-existing Kafka truststore password in `config/abs-defaults.yml`.
`management_port`	API Publish service port. Default: `8050`
`jks_password`	API Publish service JKS password. You can change the password for the JKS file. It will be generated during installation.
`mongo_certificate_verify`	Mongodb Server Certificate Verification for API Publish service. Default: `false`

Copyright © 2025 Ping Identity Corporation

Variable	Description
`server_ssl_key_alias`	Alias for API Publish service SSL JKS file. Default: `pingidentity`
`data_dbname`	API Publish service database name. Default: `abs_data`
`meta_database`	API Publish service metadatabase name. Default: `abs_metadata`
`current_admin_password`	API Publish service CLI password. Default: `admin`
`new_admin_password`	API Publish service new admin password. Default: `admin`

Changing the ABS default system memory

About this task

To change the default system memory in the `abs.properties` file of ABS:

Steps

1. Go to the `software` directory.
2. Untar the ABS binary by entering the following command:

   ```
   # tar –zxvf pi-api-abs-5.0.tar.gz
   ```

3. Edit the `config/abs.properties` file to change the default value of `system_memory` to 50% of host memory.

   ```
   # vi pingidentity/abs/config/abs.properties
   ```

   Example:

   If host ABS system has 16 GB of memory, set the value to `8192` MB.

4. Save the file.
5. Tar the ABS binary and save it with the same file name (`pi-api-abs-5.0.tar.gz`) in the `software` directory by entering the following command:

   ```
   # tar -czf pi-api-abs-5.0.tar.gz pingidentity/abs
   ```

Copyright © 2025 Ping Identity Corporation

Changing Dashboard default settings

You can change the default settings of PingIntelligence for APIs Dashboard.

To change the default settings, edit the `dashboard-defaults.yml` file and `ilm.json` file.

Changing settings in `dashboard-defaults.yml`

You can change the default settings of PingIntelligence Dashboard by editing the `/<pi-install-path>/pingidentity/pi-api-deployment/config/dashboard-defaults.yml` file.

The following table lists the variables that you can set for PingIntelligence Dashboard in various configurations.

Variable	Description
`port`	Port number to connect to PingIntelligence Dashboard.
`authentication_mode`	Defines the mode in which Dashboard authenticates. The valid values are `native` and `sso`. **Note** You should use `native` authentication for proof-of-concept deployments.
`session_max_age`	Defines the maximum time for a session. The configured values should be in the form of *<number><duration_suffix>*. Duration should be > 0. Allowed `duration_suffix` values: `m` for minutes, `h` for hours, and `d` for days.
`max_active_sessions`	Defines the maximum number of active UI sessions at any given time. The value should be greater than 1.
`admin_password` and `ping_user_password`	The passwords for webgui `admin` and `ping_user` accounts. **Note** `admin_password` and `ping_user_password` are applicable in native authentication_mode only.
Single sign-on (SSO) configurations - Applicable only when `authentication_mode` is set as `sso`	
`sso_oidc_client_id`	Client ID value in configured in the identity provider.
`sso_oidc_client_secret`	Client secret configured for the corresponding Client ID.
`sso_oidc_client_authentication_method`	OpenID Connect (OIDC) client authentication mode. The valid values are `BASIC`, `POST`, or `NONE`

Copyright © 2025 Ping Identity Corporation

Variable	Description
`sso_oidc_provider_issuer_uri`	HTTPS IP address of OIDC provider. Also, place the SSO provider's issuer-certificate in the following path: `<installation_path>/pingidentity/certs/webgui/`
`sso_oidc_provider_user_uniqueid_claim_name`	Claim name for unique ID of the user in UserInfo response. A new user is provisioned using this unique ID value.
`sso_oidc_provider_user_first_name_claim_name`	Claim name for first name of the user in UserInfo response. Either first name or last name can be empty, but both should not be empty.
`sso_oidc_provider_user_last_name_claim_name`	Claim name for last name of the user in UserInfo response. Either first name or last name can be empty, but both should not be empty.
`sso_oidc_provider_user_role_claim_name`	Claim name for role of the user in UserInfo response. The default value is `role`.
`sso_oidc_client_additional_scopes`	Additional scopes in authorization request. Multiple scopes should be comma (,) separated values. OpenID profile scopes are always requested.
End-of-SSO configurations	
SSL configuration for PingIntelligence Dashboard • `server_ssl_key_store_password` • `server_ssl_key_alias`	Configure the passwords for key store and key alias.
H2 database configuration: • `h2_db_password` • `h2_db_encryption_password`	Password for H2 database and password for encryption

Copyright © 2025 Ping Identity Corporation

Variable	Description
Discovery configuration - The following variables configure discovery settings for Dashboard: • `discovery_source` • `discovery_mode` • `discovery_mode_auto_polling_interval` • `discovery_mode_auto_delete_non_discovered_apis` Discovery source - Defines the details of discovery source for PingAccess or Axway API gateway. PingAccess: • `pingaccess_url` • `pingaccess_username` • `pingaccess_password` Axway • `axway_url` • `axway_username` • `axway_password`	• `discovery_source` - Defines the source of discovered APIs. The discovery source can be `abs`, `pingaccess`, or `axway` • `discovery_mode` - Defines the mode in which Dashboard publishes APIs to ASE. It can either `auto` or `manual` mode. For more information on discovery mode, seeDiscovered APIs • `discovery_mode_auto_polling_interval` - If the mode is set to `auto` in previous option, then configure the time interval in minutes for publishing the APIs to ASE. It recommended to keep a minimum time interval of 10 minutes. • `discovery_mode_auto_delete_non_discovered_apis` - If the mode is set to `auto`, you can configure whether you want to delete the other APIs from ASE when Dashboard publishes the discovered APIs. Configure PingAccess or Axway URL, username and password if the discovery source is `pingaccess` or `axway`.
`enable_xpack`	Configures whether the deployment package installs Xpack. The default value is `true`. If you are using an existing Elasticsearch and authentication is not configured for Xpack, set `enable_xpack` to `false`.
`elasticsearch_url`	If you have set `install_elasticsearch` as `false` in the `hosts` file, configure the Elasticsearch Uniform Resource Locator (URL). Enter the complete URL, including `http/https`. For example, https://myelasticsearchurl.pi.com:443. NOTE: Providing the port number in the URL is mandatory.
`elasticsearch_distro_type`	Configure the distribution type of Elasticsearch. Allowed values are `default` or `aws`. **Note** This variable is available for configuration in PingIntelligence for APIs 4.4.1.
`elastic_username`	If you want to use an already available Elasticsearch username, configure it in `elastic_username`.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`elastic_password`	Elasticsearch password. The default value is `changeme`. **Note** Do not change the `elastic_password` after PingIntelligence installation is complete.
`elasticsearch_purge_schedule`	The schedule for Elasticsearch purge to run.
`elasticsearch_purge_days`	The number of days for Elasticsearch purge.
`consumer_user`	Consumer user in Kafka. Default: `pi4api_de_user`
`consumer_authentication_password`	Consumer user password. Default: `changeme`
`dataengine_groupid`	Group in Kafka for data engine consumer. Default: `pi4api.data-engine`
`ping_user_password`	Password for the default user name `ping_user`.
`ping_admin_password`	Password for the admin.
Syslog configuration: • `enable_syslog` • `host, port` • `facility`	Configure `syslog` details. Setting `enable_syslog` to `true` lets dashboard engine log the ABS detected attacks in the `syslog` server. Provide the host and port number of the `syslog` server.

Important

Make a backup of the `dashboard-defaults.yml` file on a secure machine after the automated installation is complete.

The following is a sample `dashboard-defaults.yml` file.

Copyright © 2025 Ping Identity Corporation

```
---
webgui:
 # Define ports for PingIntelligence WebGUI
 # Make sure ports are not same for single server installation
 port: 8030

 # allowed values: native, sso.
 # In native mode, webgui users are self managed and stored in webgui.
 # In sso mode, webgui users are managed and stored in an Identity provider.
 authentication_mode: native
 # Maximum duration of a session.
 # Value should be in the form of <number><duration_suffix>
 # Duration should be > 0.
 # Allowed duration_suffix values: m for minutes, h for hours, d for days.
 session_max_age: 6h

 # Number of active UI sessions at any time.
 # Value should be greater than 1.
 max_active_sessions: 50

  admin_password and ping_user_password are applicable in native authentication_mode only.
 # webgui "admin" account password
 admin_password: changeme
 # webgui "ping_user" account password
 ping_user_password: changeme

  Below sso configuration properties are applicable in sso authentication_mode only.
 # Client ID value in Identity provider.
 sso_oidc_client_id: pingintelligence
 # Client Secret of the above Client ID.
 sso_oidc_client_secret: changeme
 # OIDC Client authentication mode.
 # Valid values: BASIC, POST, or NONE
 sso_oidc_client_authentication_method: BASIC
 # OIDC Provider uri
 # WebGUI queries <issuer-uri>/.well-known/openid-configuration to get OIDC provider metadata
 # issuer ssl certificate is not trusted by default. So import issuer ssl certificate into config/
webgui.jks
 # issuer should be reachable from both back-end and front-end
 sso_oidc_provider_issuer_uri: https://127.0.0.1:9031

 # Place the sso provider issuer-certificate in the following path => <installation_path>/pingidentity/
certs/webgui/
 # Name of the file should be => webgui-sso-oidc-provider.crt

 # claim name for unique id of the user in UserInfo response
 # a new user is provisioned using this unique id value
 sso_oidc_provider_user_uniqueid_claim_name: sub
 # claim name for first name of the user in UserInfo response
 # either first name or last name can be empty, but both should not be empty
 sso_oidc_provider_user_first_name_claim_name: given_name
 # claim name for last name of the user in UserInfo response
 # either first name or last name can be empty, but both should not be empty
 sso_oidc_provider_user_last_name_claim_name: family_name
```


Copyright © 2025 Ping Identity Corporation

```
  # claim name for role of the user in UserInfo response
  sso_oidc_provider_user_role_claim_name: role
  # additional scopes in authorization request
  # multiple scopes should be comma (,) separated
  # openid,profile scopes are always requested
  sso_oidc_client_additional_scopes:
  ## End of sso configuration

  # ssl key store password of webgui hosts
  server_ssl_key_store_password: changeme
  server_ssl_key_alias: webgui

  # local h2 db datasource properties
  h2_db_password: changeme
  h2_db_encryption_password: changeme

  # allowed values: abs/pingaccess/axway
  discovery_source: abs
  # allowed values: auto/manual
  discovery_mode: auto
  # value is in minutes
  discovery_mode_auto_polling_interval: 10
  discovery_mode_auto_delete_non_discovered_apis: false

  # valid only if discovery_source is set to pingaccess
  pingaccess_url: https://127.0.0.1:9000/
  pingaccess_username: Administrator
  pingaccess_password:

  # valid only if discovery_source is set to axway
  axway_url: https://127.0.0.1:8075/
  axway_username: apiadmin
  axway_password:


dataengine:
  ui:
    # Install elasticsearch with xpack enabled
    # If there is no authentication on pre-existing elasticsearch, set this to false
    enable_xpack: true
    server_port: 8040
    # When install_elasticsearch is set to false in config/hosts, this url will be used
    # Give the complete url with https/http and elasticsearch port number
    # Make sure elasticsearch_url is accessible from ansible management host, dataengine, webgui nodes.
    elasticsearch_url: https://search-giueibohzd6pfijfysjfsxucty.pingidentity.com:443
    # Configuration distribution type of elasticsearch. Allowed values are default or aws
    elasticsearch_distro_type: default

    # User with permission set similar to "elastic" user
    elastic_username: elastic

    # Passwords for "elasticsearch","ping_user" and "ping_admin" users
    # dataengine will be accessible for these accounts
    # Please set strong passwords
    # If enable_xpack is set to false, below passwords are ignored
```

Copyright © 2025 Ping Identity Corporation

```
    elastic_password: changeme

     # ssl key store password of webgui hosts
    server_ssl_key_store_password: changeme
    server_ssl_key_alias: dataengine

    #Users ,passowrd and groupid for dataengine in kafka
    consumer_user: pi4api_de_user
    consumer_authentication_password: changeme
    dataengine_groupid: pi4api.data-engine

    #Elastic Search Purge Schedule
    elasticsearch_purge_schedule: "0 23 * * * * "
    elasticsearch_purge_days: "30"

  syslog:
    # Configuration for syslog
    enable_syslog: false
    host: localhost
    port: 614
    facility: LOCAL0
```

Changing settings in `ilm.json`

You can change the default settings of Index Lifecycle Management (ILM) policy by editing the `/<pi-install-path>/pingidentity/pi-api-deployment/config/ilm.json` file.

The ILM policy allows you to manage the lifecycle of the Elasticsearch indices. The following table lists the variables that you can set in the `ilm.json` file. For more information on `ilm.json` configuration, see Automatic rollover index.

Variable	Description
`max_size`	Defines the maximum size of the Elasticsearch rollover index. When the index size reaches the defined value, it rolls over. `max_size` value should be a positive non-zero number. Allowed units are MB and GB.
`max_age`	Defines the maximum age of the Elasticsearch rollover index configuration. The `max_age` value should be a positive non-zero number. Allowed units are `h` for hours and `d` for the number of days. If both `max_size` and `max_age` are configured, then the index rolls over based on the value that is achieved first.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`min_age`	Defines the minimum age, after which the Elasticsearch rollover index enters into a different phase. Allowed units are `h` for hours and `d` for the number of days. Every index starts from `hot` phase. For more information on the phases in an index life cycle, see Automatic rollover index.
`priority`	Defines the sequence in which indices are reloaded back into memory when Elasticsearch restarts. Use a positive integer number to set the priority.

Important

Rollover index configuration takes effect only when `enable_xpack` is set to `true` in the `dashboard-default.yml` file. For more information, see Changing settings in `dashboard-defaults.yml` .

The following is a sample `ilm.json` file.

```
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_size": "7GB",
            "max_age": "7d"
          },
          "set_priority": {
            "priority": 100
          }
        }
      },
      "warm": {
        "min_age": "30d",
        "actions": {
          "set_priority": {
            "priority": 50
          }
        }
      },
      "cold": {
        "min_age": "90d",
        "actions": {
          "freeze": {},
          "set_priority": {
            "priority": 0
          }
        }
      }
    }
  }
}
```

Copyright © 2025 Ping Identity Corporation

Changing Kafka and Zookeeper default settings

Kafka and Zookeeper will be installed as part of the deployment framework.

By default, in the `config/hosts` file, Kafka and Zookeeper are configured to be installed as part of the deployment framework:

```
# this option can be used if there is an existing kafka installation that can be used
# set it to false if kafka need not be installed
install_kafka=true
```

- To disable the Kafka and Zookeeper installation, set `install_kafka=false`.
- Edit the parameters in `config/kafka-defaults.yml` to change the default installation settings for Kafka and Zookeeper.

```
kafka:
 # Define ports for the Kafka brokers
 # These ports remain same for all brokers
 ssl_port: 9094
 #Port to be used for Communication with abs and dataengine
 sasl_port: 9093

 #kafka jks password
 jks_password: changeme


 #Enable Delete topics in kafka
 delete_topic: false

 ssl_key_alias: pingidentity
 wait_time_before_clean: 30
 startup_timeout: 120

zookeeper:

   # Define ports for the zookeeper brokers
   # These ports remain same for all zookeeper
   ssl_port: 2182
```

Kafka variables

Variable	Description
`ssl_port`	Secure Sockets Layer (SSL) port for Kafka. Default: 9094
`sasl_port`	Simple Authentication and Security Layer (SASL) port that is also used by the data engine and ABS to communicate with Kafka Default: 9093

Copyright © 2025 Ping Identity Corporation

Variable	Description
`jks_password`	Java KeyStore (JKS) password. If a custom truststore and key store is provided, you can configure the password here.
`delete_topic`	Enables topic deletion in Kafka.
`wait_time_before_clean`	Waiting time before cleaning existing data in Kafka, for new installation.
`startup_timeout`	Waiting time for Kafka to start.

Zookeeper variables

Variable	Description
`ssl_port`	SSL port for Zookeeper, also used for communication to Kafka via Kafka's SSL port. Default: 9093

Changing default Kafka configurations in `config/hosts`

The `config/hosts` file has a Kafka configuration, for example:

```
[kafka]
172.16.40.81 zookeeper_id=1
```

You can install Kafka and Zookeeper as a cluster by providing multiple IPs or hosts in the host file, and `zookeeper_id` according to the number of nodes to install.

Copyright © 2025 Ping Identity Corporation

Note

- `zookeeper_id` should start from 1.
- Each new node's `zookeeper_id` should increase by 1.
- The nodes should be listed in ascending `zookeeper_id` order.
- The number of nodes should only be either one or three, for example:
  - One node:

```
[kafka]
<IP address> zookeeper_id=1
```

  - Three nodes:

```
[kafka]
<IP address 1> zookeeper_id=1
<IP address 2> zookeeper_id=2
<IP address 3> zookeeper_id=3
```

You can provide custom `crt` and `key` files in the `certs` folder location:

```
kafka_private_key_location: "{{ cert_dir }}/kafka/kafka.key"
kafka_cert_location: "{{ cert_dir }}/kafka/kafka.crt"
```

Configure Kafka keystore password

You can configure the keystore password in `config/kafka-defaults.yml`. The keystore and truststore will be generated dynamically and will be used for `zookeeper-kafka` and `kafka-client` communication.

Configuring system parameters

Configure system parameters by running the command below or manually if the configured user does not have `sudo` access.

Before you begin

The following two system parameters are required to be set before installing the PingIntelligence software:

- For Elasticsearch: `vm.max_map_count`
- For API Security Enforcer (ASE), API Behavioral Security (ABS), MongoDB, and Elasticsearch: `ulimit`

Copyright © 2025 Ping Identity Corporation

Command-based configuration

Configuring command-based system parameters
Before you begin

The script in this task uses `sudo` access for the user on the Elasticsearch, ASE, ABS, and MongoDB hosts. Ensure the Internet Protocol (IP) address of these hosts was configured in the `hosts` file. See Creating a new SSH user and configuring user authentication.

About this task

To set up system parameters using command-based configuration:

Steps

1. Run the following command to configure the system parameters on the respective virtual machines (VMs).

   > **Note**
   >
   > Make sure that the following command is run only when `install_as_sudo` is set to `true` in the `hosts` file.

   ```
   [pi-api-deployment]# ./bin/start.sh configure
   Please see /opt/pingidentity/pi-api-deployment/logs/ansible.log for
   more details.
   ```

   Example:

Copyright © 2025 Ping Identity Corporation

An example `ansible.log` file for a successful launch of EC2 instances is shown below:

```
[pi-api-deployment]# tail -f logs/ansible.log

================================================================================
Current Time: Sun Jun 07 06:05:25 EST 2020
Starting configure scripts
================================================================================
Sun Jun 07 06:05:25 EST 2020: Setting up local environment
Sun Jun 07 06:05:25 EST 2020: Installing packages
Sun Jun 07 06:05:25 EST 2020: Installing pip and ansible

PLAY [Configure system settings for elasticsearch] *

TASK [Get vm.max_map_count]
TASK [Set vm.max_map_count if less than 262144]
TASK [Get ulimit -n]
TASK [Set ulimit nofile to 65536 if value is low - softlimit] *
TASK [Set ulimit nofile to 65536 if value is low - hardlimit]

PLAY RECAP *
192.168.11.143             : ok=7    changed=1    unreachable=0    failed=0
192.168.11.144             : ok=3    changed=0    unreachable=0    failed=0
192.168.11.145             : ok=5    changed=2    unreachable=0    failed=0

Sun Jun 07 06:06:14 EST 2020: Configure successful
================================================================================
```


Copyright © 2025 Ping Identity Corporation

Manual configuration

Configuring system parameters manually
About this task

If the configured user does not have `sudo` access, then manually edit the `vm.max_map_count` and `ulimit` values:

Steps

1. Set the `vm.max_map_count` to `262144` on the Elasticsearch virtual machine (VM) by entering the following command:

```
$sudo sysctl -w vm.max_map_count=262144
```

2. To make the setting persistent across reboots, run the following command:

```
$sudo echo "vm.max_map_count=262144" >> /etc/sysctl.conf
```

3. Set the `ulimit` to `65536` on the ASE, ABS, MongoDB, and Elasticsearch hosts. To set the `ulimit`:
    1. Edit `/etc/security/limits.conf` for increasing the soft limit and hard limit.
    2. Add the following two lines for the user that you have created (for example, `pi-user`):

```
pi-user soft nofile 65536
pi-user hard nofile 65536
```

> **Note**
>
> If the number of APIs in the environment is greather than 1500, then set the `ulimit` to `131070`.

Installing PingIntelligence for APIs software

The automated deployment framework creates the updated package for each PingIntelligence component and stores them in the `/opt/pingidentity/pi-api-deployment/software/updated_packages` directory.

About this task

The keys, passwords, and port number in these packages are the ones that you configured using the `.yml` files in the `/opt/pingidentity/pi-api-deployment/config` directory. You can use these packages to install PingIntelligence components on other instances.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> The command-line interface (CLI) admin password in ASE and ABS is saved in the updated packages. If you want to change the CLI admin password, use the **`update_password`** command in ASE and ABS to manually update the password.

To set up the deployment:

Steps

1. Run the following command:

```
[pi-api-deployment]# ./bin/start.sh install
Please see /opt/pingidentity/pi-api-deployment/logs/ansible.log for more details.
```

2. Accept the End User License Agreement (EULA) displayed for ABS for installation to start.
3. To verify a successful setup, view the `ansible.log` file.

   Result:

   Below is a log file snippet for a successful setup:

```
[pi-api-deployment]# tail -f logs/ansible.log
================================================================================
Current Time: Sun Jun 07 06:06:22 EST 2020
Starting setup scripts
================================================================================
Sun Jun 07 06:06:22 EST 2020: Setting up local environment
Sun Jun 07 06:06:22 EST 2020: Installing packages
Sun Jun 07 06:06:23 EST 2020: Installing pip and ansible
.
.
PLAY RECAP ***
127.0.0.1                  : ok=9    changed=0    unreachable=0    failed=0
192.168.11.143             : ok=25   changed=13   unreachable=0    failed=0
192.168.11.144             : ok=57   changed=39   unreachable=0    failed=0
192.168.11.145             : ok=56   changed=35   unreachable=0    failed=0

Sun Jun 07 06:23:37 EST 2020: Setup successful
================================================================================
```

systemd service">

Installing PingIntelligence as a `systemd` service

You can install the various PingIntelligence components as a `systemd` service. When installing as a service, the various components are started automatically when the host system restarts.

Before you begin

Copyright © 2025 Ping Identity Corporation

Ensure you have `sudo` access to install PingIntelligence components as a service. Complete the following steps only if the automated deployment did not install PingIntelligence components as a service. To verify whether service is installed on the desired host machine, run the following command:

```
# systemctl status <service-name>
```

For example, to check ASE service, enter the following command on ASE host machine:

```
systemctl status pi-ase.service
● ase.service - ASE
  Loaded: loaded (/etc/systemd/system/ase.service; disabled; vendor preset: disabled)
  Active: active (running) since Sun 2019-11-03 23:01:19 MST; 23h ago
.
.
Nov 03 23:01:19 T5-06 systemd[1]: Started ASE.
```

Verify that PingIntelligence services are not running. Use the following service names to verify the status of each component:

- ASE: `pi-ase.service`
- ABS: `pi-abs.service`
- MongoDB: `pi-mongodb.service`
- Dashboard: `pi-dashboard.service`
- Web GUI: `pi-webgui.service`
- Elasticsearch: `pi-elasticsearch.service`
- Kafka: `pi-kafka.service`
- Zookeeper: `pi-zookeeper.service`
- API Publish: `pi-apipublish.service`

Stop the component for which you want to install the service.

About this task

To install PingIntelligence as a `systemd` service:

Steps

1. Sign on to the host machine for which you want to install the service.

   Example:

   If you want to install ASE as a service, sign on to the ASE host machine.

2. Navigate to the `util` directory. Enter the following command as a `root` user to install PingIntelligence as a service:

   ```
   #sudo ./install-systemctl-service.sh  <component_name>  <ansible_user_name>
   ```

Copyright © 2025 Ping Identity Corporation

Example:

On the ASE host machine, run the following command:

```
#sudo ./install-as-service.sh pi-ase pi-user
```

3. Install service for each component following steps 1 and 2 on the respective host machines.

Next steps

Edit the service files in the following order to make sure that PingIntelligence components successfully installed. Use the `Required` option to set the order of starting of service. For more information, see Creating and modifying systemd unit files.

1. MongoDB
2. Kafka
3. ABS
4. ASE
5. API Publish
6. Elasticsearch
7. Dashboard
8. Web GUI

Verifying PingIntelligence installation

Verify that all PingIntelligence components have installed and started successfully.

Verifying ASE installation

About this task

To verify ASE was successfully installed:

Steps

1. Sign on to the ASE host machine and navigate to `<installation-path>/pingidentity/ase/bin` directory.
2. Run the `status` command:

Copyright © 2025 Ping Identity Corporation

```
/home/pi-user/pingidentity/ase/bin/cli.sh status
Ping Identity Inc., API Security Enforcer
status              : started
mode                : inline
http/ws             : port 8090
https/wss           : port 8443
firewall            : enabled
abs                 : disabled, ssl: enabled
abs attack          : disabled
audit               : enabled
ase detected attack : disabled
attack list memory  : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

Result:

If the `status` command runs successfully, then ASE has been installed and started.

Verifying ABS and MongoDB installation

About this task

To verify ABS and MongoDB were successfully installed:

Steps

1. Sign on to the ABS EC2 instance and run the ABS Admin REST API using a REST API client, such as Postman.

   > **Note**
   >
   > For more information on installing and configuring Postman, see ABS Administration.

2. To access the REST API report, call the ABS system at the following URL: https://<*abs_ip*>:<*abs_port*>/v5/abs/admin. Use the IP address from the `hosts` file.

   Result:

   If ABS and MongoDB were installed successfully, the Admin REST API output will display the MongoDB nodes. If the admin API is not accessible, then ABS has not started. The following is a sample output of the Admin REST API:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, and ASE logs",
    "license_info": {
        "tier": "Free",
        "expiry": "Sun Jan 10 00:00:00 UTC 2021",
        "max_transactions_per_month": 0,
        "current_month_transactions": 30,
        "max_transactions_exceeded": false,
        "expired": false
    },
    "across_api_prediction_mode": true,
    "poc": true,
    "api_discovery": {
        "subpath_length": "1",
        "status": true
    },
    "apis": [
        {
            "api_name": "atm_app_oauth",
            "host_name": "",
            "url": "/atm_app_oauth",
            "api_type": "regular",
            "creation_date": "Thu Mar 05 08:54:01 UTC 2020",
            "servers": 1,
            "protocol": "https",
            "cookie": "JSESSIONID",
            "token": false,
            "training_started_at": "Fri Feb 14 06:44:06 UTC 2020",
            "training_duration": "1 hour",
            "prediction_mode": true,
            "apikey_header": "X-API-KEY-2",
            "apikey_qs": "",
            "jwt": {
                "username": "",
                "clientid": "",
                "location": ""
            }
        },
        {
            "api_name": "root_api",
            "host_name": "",
            "url": "/",
            "api_type": "regular",
            "creation_date": "Thu Mar 05 08:54:01 UTC 2020",
            "servers": 1,
            "protocol": "https",
            "cookie": "JSESSIONID",
            "token": false,
            "training_started_at": "n/a",
            "training_duration": "n/a",
            "prediction_mode": false,
            "apikey_header": "X-API-KEY-1",
            "apikey_qs": "",
            "jwt": {
                "username": "",
                "clientid": "",
                "location": ""
            }
```


Copyright © 2025 Ping Identity Corporation

```
        }
    ],
    "abs_cluster": {
        "abs_nodes": [
            {
                "node_ip": "127.0.0.1",
                "os": "Red Hat Enterprise Linux Server - VMware, Inc.",
                "cpu": "16",
                "memory": "31G",
                "filesystem": "3%",
                "bootup_date": "Fri Feb 28 08:13:19 UTC 2020"
            },
            {
                "node_ip": "127.0.0.1",
                "os": "Red Hat Enterprise Linux Server - VMware, Inc.",
                "cpu": "16",
                "memory": "31G",
                "filesystem": "4%",
                "bootup_date": "Tue Mar 24 06:35:47 UTC 2020"
            }
        ],
        "mongodb_nodes": [
            {
                "node_ip": "127.0.0.1:27017",
                "status": "primary"
            }
        ]
    },
    "ase_logs": [
        {
            "ase_node": "88968c39-b4ea-4481-a0b4-d0d651468ab5",
            "last_connected": "Thu Mar 05 08:40:14 UTC 2020",
            "logs": {
                "start_time": "Thu Mar 05 08:40:14 UTC 2020",
                "end_time": "Thu Mar 05 08:40:14 UTC 2020",
                "gzip_size": "0.74KB"
            }
        },
        {
            "ase_node": "e6b82ce9-afb3-431a-8faa-66f7ce2148b9",
            "last_connected": "Thu Mar 05 08:54:06 UTC 2020",
            "logs": {
                "start_time": "Thu Mar 05 08:54:06 UTC 2020",
                "end_time": "Thu Mar 05 08:54:06 UTC 2020",
                "gzip_size": "2.82KB"
            }
        },
        {
            "ase_node": "4df50c47-407a-41f9-bda6-b72dc34dadad",
            "last_connected": "Fri Feb 28 07:20:03 UTC 2020",
            "logs": {
                "start_time": "Tue Feb 25 12:50:00 UTC 2020",
                "end_time": "Fri Feb 28 07:20:03 UTC 2020",
                "gzip_size": "76.01KB"
            }
        },
        {
            "ase_node": "1910051e-5bab-44e6-8816-5b5afffdd1cf",
            "last_connected": "Tue Feb 18 08:10:05 UTC 2020",
            "logs": {
                "start_time": "Fri Feb 14 06:42:38 UTC 2020",
                "end_time": "Tue Feb 18 08:10:05 UTC 2020",
```


Copyright © 2025 Ping Identity Corporation

```
                "gzip_size": "2.89MB"
            }
        }
    ],
    "percentage_diskusage_limit": "80%",
    "scale_config": {
        "scale_up": {
            "cpu_threshold": "70%",
            "cpu_monitor_interval": "30 minutes",
            "memory_threshold": "70%",
            "memory_monitor_interval": "30 minutes",
            "disk_threshold": "70%",
            "disk_monitor_interval": "30 minutes"
        },
        "scale_down": {
            "cpu_threshold": "10%",
            "cpu_monitor_interval": "300 minutes",
            "memory_threshold": "10%",
            "memory_monitor_interval": "300 minutes",
            "disk_threshold": "10%",
            "disk_monitor_interval": "300 minutes"
        }
    },
    "attack_ttl": {
        "ids": [
            {
                "id": "ip",
                "ttl": 120
            },
            {
                "id": "cookie",
                "ttl": 120
            },
            {
                "id": "access_token",
                "ttl": 120
            },
            {
                "id": "api_key",
                "ttl": 240
            },
            {
                "id": "username",
                "ttl": 360
            }
        ]
    }
}
```


Verifying Dashboard installation

About this task

To verify Dashboard installation:

Steps

1. Enter the Dashboard IP address from the `hosts` file in your web browser.

Copyright © 2025 Ping Identity Corporation

2. Sign on using `ping_user` or `admin` as the username and the password configured in the `dashboard-defaults.yml` file. If the authentication mode is set to `SSO`, then sign on using your SSO username and password.

Next steps

See the API Security Enforcer, ABS AI Engine, and PingIntelligence Dashboard guides for configuration and administration of PingIntelligence products.

Integrating PingIntelligence into your environment

After the installation is complete, refer to the following topics based on the type of deployment.

Sideband configuration

Sideband configuration

After you have completed the deployment, integrate one of the following API gateways with PingIntelligence components and start sending the API traffic to your API gateway:

- Akana API gateway sideband integration
- Apigee integration
- AWS API Gateway integration
- Azure APIM sideband integration
- Axway sideband integration
- CA API gateway sideband integration
- F5 BIG-IP integration
- IBM DataPower Gateway sideband integration
- Kong API gateway integration
- MuleSoft sideband integration
- NGINX sideband integration
- NGINX Plus sideband integration
- PingAccess sideband integration
- PingFederate sideband integration
- WSO2 integration

Copyright © 2025 Ping Identity Corporation

Inline configuration

Inline configuration

If you configured PingIntelligence ASE as Inline ASE, the next step is to add Defining an API using API JSON configuration file in inline mode.

After adding API definitions, direct your API client to the IP address of the ASE software on port 80 or 443.

Refer to the following topics within the ASE and ABS Admin Guides:

- ASE port information
- API naming guidelines
- ASE and ABS integration

After you have added your APIs in ASE, the API model needs to be trained. The training of the API model is completed in ABS.

== Training the API model

After you have added your APIs in ASE, the API model needs to be trained. The training of the API model is completed in ABS. The following topics give a high-level overview of the training process; however, it is recommended to review the entire ABS Admin Guide:

- Train your API model
- Generate and view the REST API reports using Postman

> **Note**
>
> To access the ABS REST API reports, you will need the following information:
> - IP address: The IP address of ABS configured in the `config/hosts` file.
> - Port number: The default value is 8080. It is configured in the `abs-defaults.yml` file.
> - API name: Name of the API for which you want to generate REST API reports.
> - Date range: The date range for which you want to generate the reports.

- Accessing the PingIntelligence Dashboard

> **Note**
>
> Sign on to the PingIntelligence for APIs Dashboard using the `ping_user` username and the password that you configured during PingIntelligence installation. For more information on password configuration, see Changing Dashboard default settings. The PingIntelligence Dashboard takes approximately one hour to start showing attack information.

Shutting down the deployment

Shut down the deployment and remove all virtual machines (VMs) and data.

Copyright © 2025 Ping Identity Corporation

About this task

When you shut down the deployment, all VMs along with the data are deleted.

Steps

1. To shut down the deployment and remove all VMs and data, run the `stop.sh` command:

```
[pi-api-deployment]# ./bin/stop.sh
Please see /opt/pingidentity/pi-api-deployment/logs/ansible.log for more details.
```

2. To verify whether the deployment was successfully stopped, check the `ansible.log` file:

```
[pi-api-deployment]# tail -f logs/ansible.log
================================================================================
Current Time: Sun Jun 07 07:23:11 EST 2020
Starting stop scripts
================================================================================
Sun Jun 07 07:23:11 EST 2020: Play stop setup
PLAY RECAP ***
192.168.11.124 : ok=2 changed=1 unreachable=0 failed=0
192.168.11.145 : ok=2 changed=1 unreachable=0 failed=0
192.168.11.146 : ok=2 changed=1 unreachable=0 failed=0
192.168.11.148 : ok=2 changed=1 unreachable=0 failed=0
192.168.11.149 : ok=4 changed=3 unreachable=0 failed=0
Sun Jun 07 07:32:53 EST 2020: Stop successful
================================================================================
```

3. Manually remove the PingIntelligence component service scripts from `/etc/systemd/system/pi-*`.

Logs

The `ansible.log` file for all the stages is available in the `/opt/pingidentity/pi-api-deployment/logs` directory.

The `logs` directory also stores `hostinfo.log` file. This log file stores information about all the hosts. Every time the automated deployment is run, the `hostinfo.log` file is appended with the host information. The following is a snippet of the log file:

Copyright © 2025 Ping Identity Corporation

```
* Wed Apr 01 02:07:26 UTC 2020 *
==========================================================================================================
Hostname: ping-rhel-3
Inventory Hostname: 172.16.40.69
PI components installed on this host:
- mongodb

Date & Time: 2020-03-31 20:05:46 MDT
Timezone: MDT
Distribution: RedHat
Release: Maipo
Distribution Version: 7.6
Kernel: 3.10.0-957.10.1.el7.x86_64
Architecture: x86_64
CPU Core: 4
RAM: 15.4951171875 GB

Filesystem             Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root  530G  132G  398G  25% /
devtmpfs               7.8G     0  7.8G   0% /dev
tmpfs                  7.8G   12K  7.8G   1% /dev/shm
tmpfs                  7.8G  335M  7.5G   5% /run
tmpfs                  7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/sda1             1014M  153M  862M  16% /boot
tmpfs                  1.6G     0  1.6G   0% /run/user/988
tmpfs                  1.6G     0  1.6G   0% /run/user/1018
tmpfs                  1.6G     0  1.6G   0% /run/user/1045
==========================================================================================================
Hostname: ping-ubuntu-1
Inventory Hostname: 172.16.40.81
PI components installed on this host:
- abs
- ase
- dashboard
- kibana
- webgui

Date & Time: 2020-03-31 20:07:16 MDT
Timezone: MDT
Distribution: Ubuntu
Release: xenial
Distribution Version: 16.04
Kernel: 4.4.0-148-generic
Architecture: x86_64
CPU Core: 4
RAM: 15.6533203125 GB

Filesystem                                  Size  Used Avail Use% Mounted on
udev                                        7.9G     0  7.9G   0% /dev
tmpfs                                       1.6G  860K  1.6G   1% /run
/dev/mapper/ubuntu--1604--template--vg-root  467G  106G  343G  24% /
tmpfs                                       7.9G  140K  7.9G   1% /dev/shm
tmpfs                                       5.0M     0  5.0M   0% /run/lock
tmpfs                                       7.9G     0  7.9G   0% /sys/fs/cgroup
```


Copyright © 2025 Ping Identity Corporation

```
/dev/sda1                                    720M  108M  576M  16% /boot
cgmfs                                        100K     0  100K   0% /run/cgmanager/fs
tmpfs                                        1.6G     0  1.6G   0% /run/user/1012
tmpfs                                        1.6G     0  1.6G   0% /run/user/1005
================================================================================================
Hostname: ping-rhel-2
Inventory Hostname: 172.16.40.228
PI components installed on this host:
- elasticsearch

Date & Time: 2020-03-31 20:06:05 MDT
Timezone: MDT
Distribution: RedHat
Release: Maipo
Distribution Version: 7.6
Kernel: 3.10.0-957.10.1.el7.x86_64
Architecture: x86_64
CPU Core: 4
RAM: 15.5126953125 GB

Filesystem             Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root  488G  7.5G  481G   2% /
devtmpfs               7.8G     0  7.8G   0% /dev
tmpfs                  7.8G   80K  7.8G   1% /dev/shm
tmpfs                  7.8G  801M  7.0G  11% /run
tmpfs                  7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/sda1             1014M  153M  862M  16% /boot
tmpfs                  1.6G     0  1.6G   0% /run/user/1015
tmpfs                  1.6G     0  1.6G   0% /run/user/1040
================================================================================================
* Wed Apr 01 02:07:26 UTC 2020 *
* Wed Apr 01 02:08:13 UTC 2020 *
================================================================================================
Hostname: ping-ubuntu-1
Inventory Hostname: 172.16.40.81
PI components installed on this host:
- abs
- ase
- dashboard
- elasticsearch
- kibana
- mongodb
- webgui

Date & Time: 2020-03-31 20:08:10 MDT
Timezone: MDT
Distribution: Ubuntu
Release: xenial
Distribution Version: 16.04
Kernel: 4.4.0-148-generic
Architecture: x86_64
CPU Core: 4
RAM: 15.6533203125 GB

Filesystem                                   Size  Used Avail Use% Mounted on
```

Copyright © 2025 Ping Identity Corporation

```
udev                                            7.9G     0  7.9G   0% /dev
tmpfs                                           1.6G  860K  1.6G   1% /run
/dev/mapper/ubuntu--1604--template--vg-root  467G  106G  343G  24% /
tmpfs                                           7.9G  140K  7.9G   1% /dev/shm
tmpfs                                           5.0M     0  5.0M   0% /run/lock
tmpfs                                           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/sda1                                       720M  108M  576M  16% /boot
cgmfs                                           100K     0  100K   0% /run/cgmanager/fs
tmpfs                                           1.6G     0  1.6G   0% /run/user/1012
tmpfs                                           1.6G     0  1.6G   0% /run/user/1005
==============================================================================================
* Wed Apr 01 02:08:13 UTC 2020 *
```

PingIntelligence manual deployment

Learn about PingIntelligence components, the different users that can install the product, and the time zone in which the components can be deployed.

Key components

PingIntelligence for APIs software combines real-time security and AI analytics to detect, report, and block cyberattacks on data and applications exposed via APIs. The software consists of three components: API Security Enforcer (ASE), API Behavioral Security (ABS) AI engine, and PingIntelligence Dashboard.

ASE

Applies real-time API metadata ingestion and enforces optional blocking. ASE can be deployed in inline or sideband mode and works with the ABS engine to identify attacks. For more information, see Inline ASE and Sideband ASE.

ABS AI Engine

Executes AI algorithms to detect in near real-time cyberattacks targeting data, applications, and systems via APIs. Attack information can be automatically pushed to all ASEs to block ongoing breaches and prevent reconnection.

PingIntelligence for APIs Dashboard

Offers you the following:

- Visibility into API activity
- View the training status and other information of your APIs
- Manage API discovery using automatic or manual mode
- View attack insight to understand why a client was flagged for an attack
- Manage attacks by unblocking clients or tune AI engine thresholds
- View ABS license information

The Dashboard engine utilizes Elasticsearch and Kibana to provide a graphical view of an API environment including user activity, attack information, and blacklisted clients.

Copyright © 2025 Ping Identity Corporation

Administrators

You can install all the PingIntelligence components either as a user with `sudo` access or a normal user (without `sudo` access). Make sure that the entire deployment is a homogenous deployment. Either all the components should be installed as a `sudo` user or as a normal user.

Time zone

All PingIntelligence components (ASE, ABS AI Engine, and Dashboard) should be installed using the same time zone, either local or UTC. MongoDB should also be configured to the same time zone as PingIntelligence components.

Installing and configuring Kafka and Zookeeper

PingIntelligence uses Kafka and Zookeeper for processing event streaming.

About this task

> **Note**
>
> From PingIntelligence 5.2, you can configure Kafka in Secure Sockets Layer (SSL) mode only.
> For more information on Kafka, see the following documentation:
>
> - .apache.org/documentation///[Kafka Introduction]
> - .apache.org/documentation///[Kafka Security Overview]

Steps

1. Create a truststore and keystore:
    1. Create `.crt` and `.key` files:

        ```
        #openssl req -new -x509 -keyout pi4api-kafka-key.key -out pi4api-kafka-crt.crt -days 730
        ```

    2. Create a `.p12` file:

        ```
        #openssl pkcs12 -export -in pi4api-kafka-crt.crt -inkey pi4api-kafka-key.key -name
        pingidentity -out kafka.p12 -password pass:changeme
        ```

    3. Create a truststore:

        ```
        #keytool -keystore kafka_truststore.jks -alias pingidentity -import -file pi4api-kafka-crt.crt
        -storepass changeme -noprompt
        ```

    4. Create a keystore:

Copyright © 2025 Ping Identity Corporation

```
#keytool -importkeystore -deststorepass changeme -deststoretype JKS -destkeystore
kafka_keystore.jks -srckeystore kafka.p12 -srcstoretype PKCS12 -srcstorepass changeme -
noprompt
```

2. Configure and start the Zookeeper service:

    1. Customize the `zookeeper.properties` file for your installation.

    Example:

```
dataDir=/home/pi-user/pingidentity/kafka/data/zookeeper
dataLogDir=/home/pi-user/pingidentity/kafka/datalog
tickTime=2000
initLimit=5
syncLimit=2
autopurge.snapRetainCount=3
autopurge.purgeInterval=0
maxClientCnxns=60
standaloneEnabled=true
admin.enableServer=true
admin.serverPort=9090
server.1=172.16.40.244:2888:3888
# the port at which the clients will connect
secureClientPort=2182

authProvider.x509=org.apache.zookeeper.server.auth.X509AuthenticationProvider
serverCnxnFactory=org.apache.zookeeper.server.NettyServerCnxnFactory
ssl.trustStore.location=/home/pi-user/pingidentity/kafka/kafka_truststore.jks
ssl.trustStore.password=changeme
ssl.keyStore.location=/home/pi-user/pingidentity/kafka/kafka_keystore.jks
ssl.keyStore.password=changeme
ssl.clientAuth=need
ssl.hostnameVerification=false
sslQuorum=true
ssl.quorum.keyStore.location=/home/pi-user/pingidentity/kafka/kafka_keystore.jks
ssl.quorum.keyStore.password=changeme
ssl.quorum.trustStore.location=/home/pi-user/pingidentity/kafka/kafka_truststore.jks
ssl.quorum.trustStore.password=changeme
ssl.quorum.hostnameVerification=false
portUnification=false
```

    2. Start the Zookeeper service:

```
#./bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

    3. Check the Zookeeper logfile:

```
#tail -f logs/zookeeper.out
```


Copyright © 2025 Ping Identity Corporation

3. Configure and start the Kafka server:

    1. Configure the SASL SCRAM server authentication file:

    ```
    vim /home/pi-user/pingidentity/kafka/config/sasl_server.conf

    KafkaServer {
            org.apache.kafka.common.security.scram.ScramLoginModule required;
    };
    ```

    2. Export the server authentication filepath as the environment variable `KAFKA_OPTS` in the Kafka server startup script `kafka-server-start.sh`.

    Example:

    ```
    #vim /bin/kafka-server-start.sh

    export KAFKA_OPTS="-Djava.security.auth.login.config=/home/pi-user/pingidentity/kafka/config/
    sasl_server.conf"
    ```

    3. Customize the `kafka/config/server.properties` file for your installation.

    Example:

Copyright © 2025 Ping Identity Corporation

```
broker.id=0
listeners=SSL://172.16.40.244:9091,SCRAM_SASL_SSL://172.16.40.244:9093
advertised.listeners=SSL://172.16.40.244:9091,SCRAM_SASL_SSL://172.16.40.244:9093
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600

log.dirs=/home/pi-user/pingidentity/kafka/data/kafka/

num.partitions=1

num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
zookeeper.connect=172.16.40.244:2182 (Important to change the SSL port)
zookeeper.connection.timeout.ms=18000
group.initial.rebalance.delay.ms=0

Appending the following

ssl.keystore.location=/home/pi-user/pingidentity/kafka/kafka_keystore.jks
ssl.keystore.password=changeme
ssl.key.password=changeme
ssl.truststore.location=/home/pi-user/pingidentity/kafka/kafka_truststore.jks
ssl.truststore.password=changeme
ssl.client.auth=required
sasl.enabled.mechanisms=SCRAM-SHA-512
ssl.enabled.protocols=TLSv1.2
listener.security.protocol.map= SSL:SSL,SCRAM_SASL_SSL:SASL_SSL
delete.topic.enable=False
authorizer.class.name=kafka.security.authorizer.AclAuthorizer
allow.everyone.if.no.acl.found=true
ssl.endpoint.identification.algorithm=
security.inter.broker.protocol=SSL
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
zookeeper.ssl.client.enable=true
zookeeper.ssl.protocol=TLSv1.2
zookeeper.ssl.truststore.location=/home/pi-user/pingidentity/kafka/kafka_truststore.jks
zookeeper.ssl.truststore.password=changeme
zookeeper.ssl.keystore.location=/home/pi-user/pingidentity/kafka/kafka_keystore.jks
zookeeper.ssl.keystore.password=changeme
zookeeper.ssl.quorum.hostnameVerification=false
zookeeper.ssl.hostnameVerification=false
zookeeper.ssl.endpoint.identification.algorithm=
```

4. Start the Kafka server:

Copyright © 2025 Ping Identity Corporation

```
#./bin/kafka-server-start.sh -daemon config/server.properties
```

5. Check the Kafka server logfile and server status:

```
# tail -f logs/kafkaServer.out
#netstat -tupln | grep -E 9093
```

4. Configure topics and access control lists (ACL) in Kafka's `config/client.properties` file.

Example:

```
# vim config/client.properties

security.protocol=SSL
ssl.truststore.location=/home/pi-user/pingidentity/kafka/kafka_truststore.jks
ssl.truststore.password=changeme
ssl.keystore.location=/home/pi-user/pingidentity/kafka/kafka_keystore.jks
ssl.keystore.password=changeme
ssl.key.password=changeme
ssl.enabled.protocols=TLSv1.2
ssl.truststore.type=JKS
ssl.keystore.type=JKS
enable.ssl.certificate.verification=false
ssl.endpoint.identification.algorithm=
```

5. Configure producer and consumer users in Zookeeper's `config/zookeeper_client.properties` file.

Example:

```
# vim config/zookeeper_client.properties

zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
zookeeper.ssl.client.enable=true
zookeeper.ssl.protocol=TLSv1.2

#zookeeper.ssl.quorum.hostnameVerification=false
#zookeeper.ssl.hostnameVerification=false
zookeeper.ssl.truststore.location=/home/pi-user/pingidentity/kafka/kafka_truststore.jks
zookeeper.ssl.truststore.password=changeme
zookeeper.ssl.keystore.location=/home/pi-user/pingidentity/kafka/kafka_keystore.jks
zookeeper.ssl.keystore.password=changeme
zookeeper.ssl.endpoint.identification.algorithm=
zookeeper.ssl.hostnameVerification=false
```

6. Create topics:

Command line and parameters:

Copyright © 2025 Ping Identity Corporation

```
<installation path>/pingidentity/kafka/bin/kafka-topics.sh
--bootstrap-server  <Kafka master IP>:<Kafka SSL port>
--create
  --topic  <ABS transactions topic>
  --partitions  <ABS topic partitions>
  --replication-factor  <ABS replication factor>
  --command-config <installation path>/pingidentity/kafka/config/client.properties
```

1. Create the transactions topic for events related to all API traffic.

   Example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-topics.sh --bootstrap-server 172.16.40.244:9091 --
   create --topic pi4api.queuing.transactions --partitions 1 --replication-factor 1 --command-
   config /home/pi-user/pingidentity/kafka/config/client.properties
   ```

2. Create the indicators of attack (IoA) topic for IoA-related events.

   Example:

   For example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-topics.sh --bootstrap-server 172.16.40.244:9091 --
   create --topic pi4api.queuing.ioas --partitions 1 --replication-factor 1 --command-config /
   home/pi-user/pingidentity/kafka/config/client.properties
   ```

3. Create the anomalies topic for anomaly-related events.

   Example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-topics.sh --bootstrap-server 172.16.40.244:9091 --
   create --topic epi4api.queuing.anomalies --partitions 1 --replication-factor 1 --command-
   config /home/pi-user/pingidentity/kafka/config/client.properties
   ```

4. Create the discovery topic for discovery-related events.

   Example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-topics.sh --bootstrap-server 172.16.40.244:9091 --
   create --topic pi4api.queuing.apis --partitions 1 --replication-factor 1 --command-config /
   home/pi-user/pingidentity/kafka/config/client.properties
   ```

7. Create users:

   Command line and parameters:

Copyright © 2025 Ping Identity Corporation

```
<installation path>/pingidentity/kafka/bin/kafka-configs.sh
--zookeeper  <Kafka master IP>:<Zookeeper.ssl_port>
--alter
  --add-config SCRAM-SHA-512=<user authentication password>
  --entity-type users
  --entity-name  <username>  -zk-tls-config-file  <installation path>/pingidentity/kafka/config/
zookeeper_client.properties
```

1. Create the ABS producer user for sending machine-learning data.

   Example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-configs.sh --zookeeper 10.96.6.126:2182 --alter --
   add-config SCRAM-SHA-512=[iterations=8192,password=changeme]] --entity-type users --entity-
   name abs_producer -zk-tls-config-file /home/pi-user/pingidentity/kafka/config/
   zookeeper_client.properties
   ```

2. Create the ABS consumer user for consuming machine-language data for job processing.

   Example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-configs.sh --zookeeper 10.96.6.126:2182 --alter --
   add-config SCRAM-SHA-512=[iterations=8192,password=changeme]] --entity-type users --entity-
   name abs_consumer -zk-tls-config-file /home/pi-user/pingidentity/kafka/config/
   zookeeper_client.properties
   ```

3. Create the data engine consumer for pulling transactions, anomalies, and indicators of compromise (IOCs).

   Example:

   ```
   /home/pi-user/pingidentity/kafka/bin/kafka-configs.sh --zookeeper 10.96.6.126:2182 --alter --
   add-config SCRAM-SHA-512=[iterations=8192,password=changeme]] --entity-type users --entity-
   name pi4api_de_user -zk-tls-config-file /home/pi-user/pingidentity/kafka/config/
   zookeeper_client.properties
   ```

8. Configure ACLs for users.

   The following table lists the topics and operations for each user type.

User	Allowed operations	Topics
ABS producer	◦ Create ◦ Read	◦ Transactions ◦ IoAs ◦ Anomalies
	Describe	Discovery

Copyright © 2025 Ping Identity Corporation

User	Allowed operations	Topics
	Write	◦ Transactions ◦ IoAs ◦ Anomalies ◦ Discovery
ABS consumer	Read	◦ Transactions ◦ IoAs ◦ Anomalies ◦ Discovery
	Describe	◦ Transactions ◦ Discovery
Data engine consumer	Read	◦ Transactions ◦ IoAs ◦ Anomalies ◦ Discovery
	Describe	Discovery

Command line and parameters:

```
<installation path>/pingidentity/kafka/bin/kafka-acls.sh
--bootstrap-server<Kafka master IP>:<Kafka SSL port>
--add
--allow-principal User:<username>
--operation <operation> [--operation <operation 2>] [--operation <operation n>]
--topic <topic name>
--command-config <installation path>/pingidentity/kafka/config/client.properties
```

1. Create the ACLs for the ABS producer user.

   Example:

   1. Transactions topic:

      ```
      /home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
      --add --allow-principal User:abs_producer --operation Create --operation Read --
      operation Write --topic pi4api.queuing.transactions --command-config /home/pi-user/
      pingidentity/kafka/config/client.properties
      ```

   2. IoAs topic:

Copyright © 2025 Ping Identity Corporation

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_producer --operation Create --operation Read --
operation Write --topic pi4api.queuing.ioas --command-config /home/pi-user/pingidentity/
kafka/config/client.properties
```

3. Anomalies topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_producer --operation Create --operation Read --
operation Write --topic epi4api.queuing.anomalies --command-config /home/pi-user/
pingidentity/kafka/config/client.properties
```

4. Discovery topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_producer --operation Create --operation Read --
operation Write --topic pi4api.queuing.apis --command-config /home/pi-user/pingidentity/
kafka/config/client.properties
```

2. Create the ACLs for the ABS consumer user.

Example:

1. Transactions topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_consumer --operation Read --operation Describe --topic
pi4api.queuing.transactions --command-config /home/pi-user/pingidentity/kafka/config/
client.properties
```

2. IoAs topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_consumer --operation Read --topic pi4api.queuing.ioas
--command-config /home/pi-user/pingidentity/kafka/config/client.properties
```

3. Anomalies topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_consumer --operation Read --topic
epi4api.queuing.anomalies --command-config /home/pi-user/pingidentity/kafka/config/
client.properties
```

4. Discovery topic:

Copyright © 2025 Ping Identity Corporation

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:abs_consumer --operation Read --topic pi4api.queuing.apis
--command-config /home/pi-user/pingidentity/kafka/config/client.properties
```

3. Create the ACLs for the data engine consumer user.

Example:

1. Transactions topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:pi4api_de_user --operation Read --topic
pi4api.queuing.transactions --command-config /home/pi-user/pingidentity/kafka/config/
client.properties
```

2. IoAs topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:pi4api_de_user --operation Read --topic pi4api.queuing.ioas
--command-config /home/pi-user/pingidentity/kafka/config/client.properties
```

3. Anomalies topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:pi4api_de_user --operation Create --operation Read --
operation Write --topic epi4api.queuing.anomalies --command-config /home/pi-user/
pingidentity/kafka/config/client.properties
```

4. Discovery topic:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091
--add --allow-principal User:pi4api_de_user --operation Create --operation Read --
operation Write --topic pi4api.queuing.apis --command-config /home/pi-user/pingidentity/
kafka/config/client.properties
```

4. Add the ACLs below in Kafka if they have not already been added:

Copyright © 2025 Ping Identity Corporation

```
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.anomalies,
patternType=LITERAL):
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=GROUP, name=pi4api.abs, patternType=LITERAL):
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.ioas,
patternType=LITERAL):
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.apis,
patternType=LITERAL):
        (principal=User:abs_producer, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=WRITE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=CREATE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=CREATE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=GROUP, name=pi4api.data-engine,
patternType=LITERAL):
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.transactions,
patternType=LITERAL):
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
```

Copyright © 2025 Ping Identity Corporation

```
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
```

9. Configure ACLs for groups.

   Command line and parameters:

```
<installation path>/pingidentity/kafka/bin/kafka-acls.sh
--bootstrap-server  <Kafka master IP>:<Kafka SSL port>
--add --allow-principal User:<username>
--operation  <operation>
--group  <group ID>
--command-config  <installation path>/pingidentity/kafka/config/client.properties
```

   1. Configure permissions for the ABS consumer user belonging to the ABS consumer group to perform read operations.

      Example:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091 --add
--allow-principal User:abs_consumer --operation Read --group pi4api.abs --command-config /
home/pi-user/pingidentity/kafka/config/client.properties
```

   2. Configure permissions for the data engine consumer user belonging to the data engine consumer group to perform read operations.

      Example:

```
/home/pi-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091 --add
--allow-principal User:pi4api_de_user --operation Read --group pi4api.data-engine --command-
config /home/pi-user/pingidentity/kafka/config/client.properties
```

ABS engine installation

The ABS engine installation process is summarized below:

- Provision systems based on the queries per second (QPS)
- Install MongoDB in a replica set
- Install ABS engine
- Connect ABS engine to MongoDB

Copyright © 2025 Ping Identity Corporation

Downloading ABS AI engine software

The ABS AI engine software must be downloaded from the Ping Identity Product Downloads site.

About this task

The Ping Identity Product Downloads site has a link to the consolidated build for ABS and API Publish. Upon clicking the link, it will download the consolidated build.

> **Note**
>
> When extracting the tar, under the `pingidentity` folder, there will be two folders:
>
> - `abs`
> - `apipublish`

Steps

1. Go to the Ping Identity Product Downloads site.
2. Under PingIntelligence for APIs, click on View Now.
3. Choose the Installation Type and click Download.
4. After downloading:

   Choose from:

   - If you are installing as a root user, copy the build file to the `/opt` directory.
   - If you are installing as a non-root user, choose any other location.

Installing ABS AI engine software

You can install ABS as a root user or as a non-root user.

Before you begin

1. Install OpenJDK 11.0.2 on a 64-bit architecture machine. To verify the Java version, run the following command:

   ```
   # java -version
   ```

2. Verify the supported operating systems. PingIntelligence supports RHEL 7.9 and Ubuntu 18.04 LTS.

About this task

The example installation path assumes that you are root user. The installation works in a similar way for a non-root user.

> **Note**
>
> It is recommended to install only one instance of ABS on each machine. MongoDB should be installed on a different machine from ABS.

Copyright © 2025 Ping Identity Corporation

Steps

1. To install ABS:

   Choose from:

   - If you are installing as a root user, change the working directory to `/opt`.
   - If you are installing as a non-root user, choose any other location.

   > **Note**
   > If you are installing as a non-root user, then increase the `ulimit -n` to 65535.

2. At the command prompt, enter `# tar –zxvf <file_name>`.

   Example:`# tar –zxvf pi-api-abs-5.2.tar.gz`

Managing the ABS license

To start ABS, you need a valid PingIntelligence license.

About this task

There are two types of licenses:

- Trial license: Valid for 30 days. At the end of the trial period, ABS stops processing.
- Subscription license: Based on the peak number of transactions subscribed for per month and the duration of the license. It is good practice to configure your email before configuring the license. ABS sends an email notification to the configured email ID when the license has expired. Contact the Ping Identity Sales team for more information.

> **Note**
> - Maximum transaction set to 0: If your subscription license has zero for maximum transaction, the license has unlimited monthly transactions. Such a license expires at the end of the subscription period.
> - License expiry: When the subscription license has expired, ABS continues to run until a restart. Then, ABS requires a valid license file to start.

Steps

1. Add an ABS license.
   1. If you have not already received a license, request a license file from the Ping Identity Sales team.

      > **Note**
      > The name of the license file must be `PingIntelligence.lic`.

   2. Copy the license file to the `/opt/pingidentity/abs/config` directory and then start ABS.
2. Update an existing license.
   1. If your existing license has expired, obtain a new license from the Ping Identity Sales team.

Copyright © 2025 Ping Identity Corporation

2. Replace the license file in the `/opt/pingidentity/abs/config` directory.
3. Stop and then start ABS after the license file is updated.

3. Check the current transaction count.
    1. To view the current transaction count against your subscription transaction limit, see Admin REST API.

       The following snippet of the Admin REST API shows the license information:

```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, and ASE logs",
    "license_info": {
        "tier": "Subscription",
        "expiry": "Wed Jan 15 00:00:00 UTC 2020",
        "max_transactions_per_month": 1000000000,
        "current_month_transactions": 98723545,
        "max_transactions_exceeded": false,
        "expired": false
    }
```

Obfuscating ABS keys and passwords

Using the ABS command line interface, you can obfuscate the keys and passwords configured in `abs.properties`.

About this task

The following keys and passwords are obfuscated:

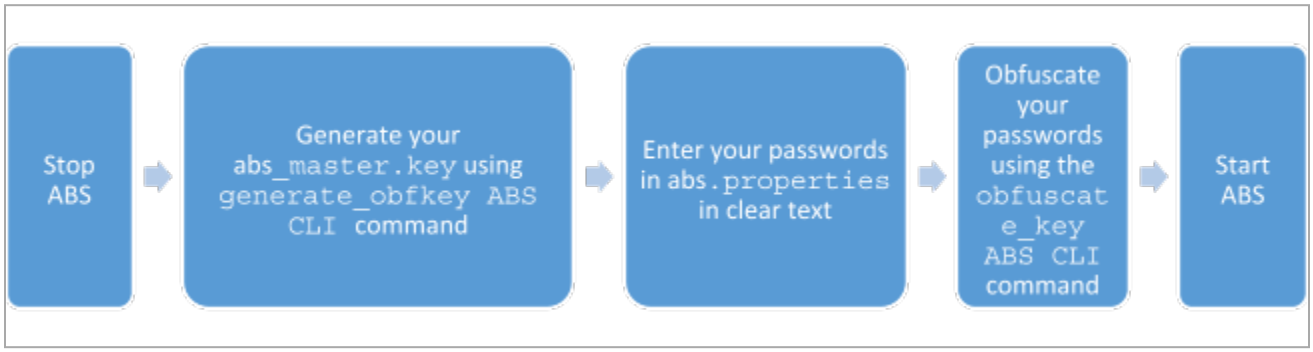
- `mongo_password`
- `jks_password`
- `email_password`

ABS ships with a default `abs_master.key`, which is used to obfuscate the various keys and passwords. It is recommended to generate your own `abs_master.key`. The default `jks_password` `abs123` is configured in the `abs.properties` file.

> **Note**
>
> During the process of obfuscation of keys and password, ABS must be stopped.

The following diagram summarizes the obfuscation process:

Copyright © 2025 Ping Identity Corporation



Steps

1. To generate the `abs_master.key`, run the `generate_obfkey` command in the ABS command-line interface (CLI):

```
/opt/pingidentity/abs/bin/cli.sh generate_obfkey -u admin -p admin

Please take a backup of config/abs_master.key before proceeding.

Warning: Once you create a new obfuscation master key, you should obfuscate all config keys also
using cli.sh -obfuscate_keys

Warning: Obfuscation master key file
/pingidentity/abs/config/abs_master.key already exist. This command will delete it create a new key
in the same file

Do you want to proceed [y/n]: y

creating new obfuscation master key
Success: created new obfuscation master key at /pingidentity/abs/config/abs_master.key
```

> **Important**
>
> In an ABS cluster, the `abs_master.key` must be manually copied to each of the cluster nodes.

Result:

The new `abs_master.key` is used to obfuscate the passwords in `abs.properties` file.

2. To obfuscate the keys and passwords:
    1. Enter the keys and passwords in clear text in `abs.properties` file.
    2. Run the `obfuscate_keys` command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/abs/bin/cli.sh obfuscate_keys -u admin -p admin

Please take a backup of config/abs.password before proceeding

Enter clear text keys and password before obfuscation.

Following keys will be obfuscated

config/abs.properties: mongo_password, jks_password and email_password
Do you want to proceed [y/n]: y

obfuscating /pingidentity/abs/config/abs.properties

Success: secret keys in /pingidentity/abs/config/abs.properties obfuscated
```

3. After passwords are obfuscated, start ABS.

> **Important**
>
> After the keys and passwords are obfuscated, the `abs_master.key` must be moved to a secure location from ABS.

Importing existing CA-signed certificates

You can import your existing CA-signed certificate in ABS.

Before you begin

To import the CA-signed certificate, you must stop ABS if it is already running.

About this task

Complete the following steps to import the CA-signed certificate:

Steps

1. Export your CA-signed certificate to PKCS12 store by entering the following command:

```
# openssl pkcs12 -export -in <your_CA_cerficate.crt> -inkey <your_certificate_key.key> -out abs.p12
-name <alias_name>
```

> **Note**
>
> If you have an intermediate certificate from CA, then append the content to the `<your_CA_certificate>.crt` file.

Example:

Copyright © 2025 Ping Identity Corporation

```
# openssl pkcs12 -export -in ping.crt -inkey ping.key -out abs.p12 -name exampleCAcertificate
Enter Export Password:
Verifying - Enter Export Password:
```

2. Import the certificate and key from the PKCS12 store to Java Keystore by entering the following command:

```
# keytool -importkeystore -destkeystore abs.jks -srckeystore abs.p12 -srcstoretype PKCS12 -alias
<alias_name> -storetype jks
```

> **Note**
>
> The command requires the destination keystore password. The destination keystore password entered in the command should be the same password that is configured in the `abs.properties` file.
> The following is a snippet of the `abs.properties` file where the destination keystore password is stored. The password is obfuscated.
>
> ```
> # Java Keystore password
> jks_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCulhAUKj7ww4=
> ```

Example:

```
# keytool -importkeystore -destkeystore abs.jks -srckeystore abs.p12 -srcstoretype PKCS12 -alias
exampleCAcertificate -storetype jks

Importing keystore abs.p12 to abs.jks...
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
```

3. Copy the `abs.jks` file created in step 2 to the `/opt/pingidentity/abs/config/ssl` directory.

4. Start ABS by entering the following command:

```
# /opt/pingidentity/abs/bin/start.sh
Starting API Behavioral Security 4.0...
please see /opt/pingidentity/abs/logs/abs/abs.log for more details
```

Next steps

> **Note**
>
> ABS supports only TLS 1.1 and TLS 1.2 and requires Open JDK 11.0.2. By default, SSL is enabled between ABS and ASE. If you need to disable SSL, contact the Ping Idenity support team.
> ABS ships with a default self-signed certificate with Java Keystore at `abs/config/ssl/abs.jks` and the default password set to abs123 in the `abs.properties` file. The default password is obfuscated in the `abs.properties` file. It is recommended to change the default passwords and obfuscate the new passwords. See Obfuscating ABS keys and passwords for steps to obfuscate passwords.

Copyright © 2025 Ping Identity Corporation

Installing MongoDB software

ABS uses a MongoDB database (5.0.18) to store analyzed logs and ABS cluster node information.

Before you begin

- Download either the RHEL or Ubuntu MongoDB 4.2 Linux `tarball` from the MongoDB website.

> **Important**
>
> The steps below use a RHEL 7 download, but the equivalent Ubuntu version of MongoDB is also supported. Use the Ubuntu MongoDB URL to download the Ubuntu version.

- Copy the following files to the MongoDB node:
  - `/opt/pingidentity/abs/mongo/abs_init.js`
  - `/opt/pingidentity/abs/mongo/abs_rs.js`

> **Important**
>
> To avoid issues in your production MongoDB deployment, it is advised to follow MongoDB recommended settings. For more information, see MongoDB Operations Checklist and MongoDB Performance.

About this task

MongoDB is installed using a replica set. In a replica set, MongoDB is installed on three nodes for high availability (HA).

> **Note**
>
> If you are installing as a non-root user, then increase the `ulimit -n` to 65535.

You can change the default username and password of MongoDB by editing the `/opt/pingidentity/abs/mongo/abs_init.js` file. Change the username and password and save the file. The following is a snippet of the `abs_init.js` file:

```
{
    user: “absuser”,
    pwd: “abs123",
    roles: [{ role: “clusterMonitor”, db: “admin” },
            { role: “readWrite”, db: “abs_metadata” },
            { role: “readWrite”, db: “abs_data” },
            { role: “readWrite”, db: “abs_mldata” },
            { role: “readWrite”, db: “local” } ]
});
```

Download MongoDB on three nodes that will form the replica set for high availability (HA) and install MongoDB on each node:

Steps

1. Create the following MongoDB directory structure on each MongoDB node:
   1. `mongo`

Copyright © 2025 Ping Identity Corporation

2. `data`
3. `logs`
4. `key`

```
# mkdir -p /opt/pingidentity/mongo/data /opt/pingidentity/mongo/logs \
/opt/pingidentity/mongo/key
```

2. Download MongoDB 5.0.18 on each node and extract to `/opt/pingidentity/mongo`.

```
# cd /opt/pingidentity/
/opt/pingidentity# wget \
https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-5.0.18.tgz \
-O mongodb.tgz && tar xzf mongodb.tgz -C /opt/pingidentity/mongo/ --strip-components=1
```

3. Update shell path variable and reload the shell.

```
/opt/pingidentity# echo PATH=$PATH:/opt/pingidentity/mongo/bin >> ~/.bashrc;
/opt/pingidentity# source ~/.bashrc
```

4. Start the MongoDB database on each node. The name of the replica set is `absrs01`. You can choose your own name for the replica set.

```
/opt/pingidentity# cd mongo
/opt/pingidentity/mongo# mongod --dbpath ./data/ --logpath ./logs/mongo.log --port 27017 --replSet
absrs01 --fork -bind_ip 0.0.0.0
```

> **Note**
>
> `bind_ip` is required for MongoDB to accept connections coming from machines other than the local host.

5. Check MongoDB connectivity among the three nodes. On MongoDB node 1, run the following command to check connectivity with node 2:

```
/opt/pingidentity/mongo# mongo --host <mongo node 2 IP address> --port 27017
```

6. Navigate to the `abs_rs.js` file and edit to configure the IP address of the primary and secondary MongoDB nodes:

Copyright © 2025 Ping Identity Corporation

```
rsconf = {
          _id: "absrs01",
          members: [
            {
             _id: 0,
             host: "127.0.0.1:27017",
             priority: 10
            },
            {
             _id: 1,
             host: "<Mongo Node 2 IP>:27017",
             priority: 2
            },
            {
             _id: 2,
             host: "<Mongo Node 3 IP>:27017",
             priority: 2
            }
           ]
        };
rs.initiate(rsconf)
rs.conf();
exit
```

> **Note**
>
> Make sure the secondary MongoDB nodes are reachable and their host names are resolvable from the primary MongoDB node.

7. Initiate the configuration by entering the following command on the shell of MongoDB node 1:

```
/opt/pingidentity/mongo# mongo --port 27017 < abs_rs.js
```

8. To verify that all the MongoDB nodes are running, enter the following on each MongoDB node:

```
 /opt/pingidentity/mongo# mongo --port 27017
```

Result:

- The primary node will display the following prompt:

  ```
  absrs01:PRIMARY>
  ```

- The secondary nodes will display the following prompt:

  ```
  absrs01:SECONDARY>
  ```

Copyright © 2025 Ping Identity Corporation

9. Create user and initialize the database using the `abs_init.js` file after making necessary modifications. On the primary node (node 1), enter the following command:

```
# mongo --host <mongo node 1 IP> --port 27017 < abs_init.js
```

> **Note**
>
> Username and password should be changed from the default values.

10. Generate a MongoDB key file.

```
/opt/pingidentity/mongo# openssl rand -base64 741 >key/mongodb-keyfile
```

11. Change the key file permission.

```
 /opt/pingidentity/mongo# chmod 600 key/mongodb-keyfile
```

12. Copy the key file generated in step 11 on each node of the replica set.
13. Shutdown MongoDB using the following command:

```
# mongod --dbpath ./data --shutdown
```

14. Restart all the MongoDB nodes with a key file and enable MongoDB authentication.

```
/opt/pingidentity/mongo# mongod --auth --dbpath ./data/ --logpath \
./logs/mongo.log --port 27017 --replSet absrs01 --fork --keyFile ./key/mongodb-keyfile -bind_ip
0.0.0.0
```

> **Note**
>
> - `bind_ip` is required for MongoDB to accept connections coming from machines other than the local host.
> - The MongoDB cache size should be restricted to 25% of system memory. You can configure this by using MongoDB's `wiredTigerCacheSizeGB` option.

Starting MongoDB with SSL

You can start MongoDB with SSL by using either a CA-signed or self-signed certificate.

About this task

To start MongoDB with SSL:

Copyright © 2025 Ping Identity Corporation

Steps

1. Add a certificate.

   Choose from:

   - To add a CA-signed certificate, create a new PEM file by concatenating the certificate and its private key. Copy the resulting PEM file to the `/opt/pingidentity/mongo/key/` directory created in step 1 in Installing MongoDB software.

     ```
     cat mongo-node-private-key mongo-node-certificate > /opt/pingidentity/mongo/key/mongodb.pem
     ```

   - To use a self-signed certificate, complete the following steps:

     1. Change directory to `key` directory:

        ```
        cd /opt/pingidentity/mongo/key
        ```

     2. Generate a self-signed certificate and key:

        ```
        openssl req -newkey rsa:2048 -new -x509 -days 365 -nodes -out mongodb-cert.crt -
        keyout mongodb-cert.key
        ```

     3. Concatenate the certificate and the key:

        ```
        cat mongodb-cert.key mongodb-cert.crt > mongodb.pem
        ```

2. After either a CA-signed certificate or self-signed certificate has been added to the `key` directory, shut down MongoDB:

   ```
   # mongod --dbpath ./data --shutdown
   ```

3. Restart MongoDB with the `-tlsMode` flag:

   ```
   mongod --auth --dbpath ./data/ --logpath ./logs/mongo.log --port 27017 --replSet absrs01 --fork --
   keyFile ./key/mongodb-keyfile -bind_ip 0.0.0.0 --tlsMode requireTLS --tlsCertificateKeyFile ./key/
   mongodb.pem
   ```

   Note

   The `-tlsMode` flag can take the following three values:
   - `allowTLS`
   - `preferTLS`
   - `requireTLS`

Next steps

Copyright © 2025 Ping Identity Corporation

Learn more in MongoDB documentation.

Changing default settings

It is recommended that you change the default key and password in ABS.

About this task

The following is a list of commands to change the default values.

Changing default JKS password

You can change the default password for the key store and key.

Before you begin

Make sure that ABS is stopped before changing the JKS password.

About this task

Complete the following steps to change the default JKS passwords. IMPORTANT: The key store and key password should be the same.

Steps

1. To change the key store password, enter the following command to change the key store password:

```
# keytool -storepasswd -keystore config/ssl/abs.jks
Enter keystore password:  abs123
New keystore password: newjkspassword
Re-enter new keystore password: newjkspassword
```

> **Note**
>
> The default key store password is `abs123` .

2. To change the key password, enter the following command to change the key password. The default key password is `abs123` .

```
# keytool -keypasswd -alias pingidentity -keypass abs123 -new newjkspassword -keystore config/ssl/
abs.jks
Enter keystore password: newjkspassword
```

> **Note**
>
> The default key password is `abs123` .

Next steps

Copyright © 2025 Ping Identity Corporation

Start ABS after you have changed the default passwords.

Changing `abs_master.key`

Create your own ABS master key to obfuscate keys and password in ABS.

Before you begin

ABS must be stopped before creating a new `abs_master.key`.

About this task

To create your own ABS master key:

Steps

1. Run the following command: `generate_obfkey`.
2. If ABS is running, then stop ABS before generating a new ABS master key. Enter the following command to stop ABS:

```
# /opt/pingidentity/abs/bin/stop.sh
checking API Behavioral Security status
sending shutdown signal to ABS, please wait...
API Behavioral Security stopped
```

3. To change `abs_master.key`, enter the `generate_obfkey` command to change the default ABS master key:

```
/opt/pingidentity/abs/bin/cli.sh generate_obfkey -u admin -p admin
Please take a backup of config/abs_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate all config keys also
using cli.sh -obfuscate_keys
Warning: Obfuscation master key file
/pingidentity/abs/config/abs_master.key already exists. This command will delete it and create a new
key in the same file
Do you want to proceed [y/n]: y
Creating new obfuscation master key
Success: created new obfuscation master key at /pingidentity/abs/config/abs_master.key
```

Changing CLI admin password

You can change the default admin password.

About this task

To change the CLI admin password:

Copyright © 2025 Ping Identity Corporation

Steps

1. Enter the following command:

```
/opt/pingidentity/abs/bin/cli.sh update_password -u admin -p admin
New Password>
Reenter New Password>
Success. Password updated for CLI
```

Changing default access and secret key in MongoDB

You can change default access and the secret key in MongoDB.

About this task

To change the default access and secret key, complete the following steps.

> **Note**
>
> ":" (colon) is a restricted character and not allowed in the access key or secret key.

Steps

1. Connect to MongoDB by entering the following command:

```
mongo --host <mongo-host> --port <mongo-port> --authenticationDatabase admin -u absuser -p abs123
```

> **Note**
>
> `absuser` and `abs123` are the default username and password for MongoDB.

2. On the MongoDB prompt, run the following command:

```
use abs_metadata
db.auth_info.updateOne( { access_key: "<new-access-key>", secret_key: "<new-secret-key>"} )
```

Connecting ABS to MongoDB

Connect ABS to MongoDB.

Steps

1. Check and open the MongoDB default port.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> The MongoDB default port for connection with ABS is 27017.

1. Run the `check_ports_abs.sh` script on the ABS machine to determine whether the default port is available.
2. Input the MongoDB host IP address and default port as arguments.

   Example:

   ```
   /opt/pingidentity/abs/util ./check_ports_abs.sh {MongoDB IPv4:[port]}
   ```

3. Run the script for the MongoDB master and slave. If the default ports are not accessible, open the port from the MongoDB machine.

2. Configure ABS to connect to MongoDB.
   1. Edit `abs_init.js` in the `/opt/pingidentity/mongo` directory to set the key values.

      Example:

      Below is a sample `abs_init.js` file:

      ```
      db.auth_info.insert({
      "access_key" : "abs_ak",
      "secret_key" : "abs_sk"
      });
      ```

      > **Note**
      >
      > ":" (colon) is a restricted character and not allowed in the access key or secret key.

      > **Note**
      >
      > Do not edit the `abs_init.js` file for any subsequent changes to the ABS access key and secret key. It is recommended to use the **`update_keys`** CLI command to change the keys. For more information, see ABS CLI commands.

   2. Copy the `abs_init.js` file from the ABS `/opt/pingidentity/abs/mongo`

      `folder to the MongoDB system [.filepath]``/opt/pingidentity/mongo`` folder.`

   3. At the MongoDB command prompt, update the MongoDB settings with the latest `abs_init.js` file.

      ```
      # mongo admin -u absuser -p abs123 < /opt/pingidentity/abs/mongo/abs_init.js
      MongoDB Shell version 5.0.18
      connecting to: admin
      switched to db abs_metadata
      WriteResult({ “nInserted” : 1})
      bye
      ```

Copyright © 2025 Ping Identity Corporation

3. Optional: To verify MongoDB SSL certificates, configure ABS to verify the validity of the MongoDB server certificate.
    1. Set the `mongo_certificate` parameter in the `/<pi_install_path>/pingidentity/abs/config/abs.properties` file.

> **Note**
>
> For more information, see Verify MongoDB SSL certificates.

Starting and stopping ABS

Learn how to start and stop ABS.

Before you begin

For ABS to start, the `abs_master.key` must be present in the `/opt/pingidentity/abs/config` directory. If you have moved the master key to a secured location for security reasons, copy it to the `config` directory before starting ABS.

About this task

You can start ABS in one of the following two ways:

- Using service script available in the `util` directory
- Using the `start.sh` script available in the `bin` directory

Steps

1. Start ABS.

    Choose from:

    - To start ABS as a service:
        1. Navigate to the `util` directory and run the following command to install ABS as a service:

```
#sudo ./install-systemctl-service.sh pi-abs
```

        2. Start the service by entering the following command:

```
systemctl start pi-abs.service
```

    - To start ABS using the `start.sh` script:
        1. Run the `start.sh` script located in the `/opt/pingidentity/abs/bin` directory.
        2. Change the working directory to `/opt/pingidentity/abs/bin`.
        3. Start ABS by entering the following command:

Copyright © 2025 Ping Identity Corporation

```
$ /opt/pingidentity/abs/bin/start.sh
Starting API Behavioral Security 4.2...
please see /opt/pingidentity/abs/logs/abs/abs.log for more details
```

4. To verify ABS has started, change the working directory to the `data` directory and look for two `.pid` files, `abs.pid`, and `stream.pid`. Also check the newly added ABS node is connecting to MongoDB and has a heartbeat.

```
> use abs_metadata
switched to db abs_metadata
> db.abs_cluster_info.find().pretty()
 {
 "_id" : ObjectId("58d0c633d78b0f6a26c056ed"),
 "cluster_id" : "c1",
 "nodes" : [
     {
         "os" : "Red Hat Enterprise Linux Server release 7.6 (Maipo)",
         "last_updated_at" : "1490088336493",
         "management_port" : "8080",
         "log_port" : "9090",
         "cpu" : "24",
         "start_time" : "1490077235426",
         "log_ip" : "2.2.2.2",
         "uuid" : "8a0e4d4b-3a8f-4df1-bd6d-3aec9b9c25c1",
         "dashboard_node" : false,
         "memory" : "62G",
         "filesystem" : "28%"
 } ] }
```

2. Stop ABS.

Choose from:

- To stop ABS using `stop.sh` script:

  1. Stop ASE (if it is running) or turn off the ABS flag in ASE.
  2. If no machine learning jobs are processing, run the `stop.sh` script available in the `bin` directory.

```
# /opt/pingidentity/abs/bin/stop.sh
checking API Behavioral Security status
sending shutdown signal to ABS, please wait...
API Behavioral Security stopped
```

> **Note**
>
> If you have started ABS as a service and try to stop it using the `stop.sh` script, ABS will restart after stopping.

Copyright © 2025 Ping Identity Corporation

- To stop ABS using the service script:

  1. Run the following command:

```
systemctl stop pi-abs.service
```

ASE installation

This section summarizes the key processes involved in API Enforcer Enforcer (ASE) installation.

A new ASE installation involves the following steps:

1. Understanding and determining ASE deployment modes.
2. Provisioning the host system based on the number of APIs and the expected queries per second (QPS).

   For information on sizing, contact the Ping Identity support team.
3. Reviewing port requirements.
4. Installing ASE.
5. Copying ASE license.
6. Changing default settings.
7. Obfuscating keys and passwords.
8. Tuning host system for high performance.
9. Starting ASE.
10. Configuring SSL for client-side connection or external APIs

> **Note**
>
> ASE cluster setup facilitates high availability and increases both performance and overall system throughput. It is recommended to set up a cluster of ASE nodes for production environments. For more information on setting up an ASE cluster, see Setting up an ASE cluster (optional) and Administering an ASE cluster.

After Installation

After installing ASE, proceed with the following tasks:

- Configuring ASE: ASE system-level configuration entails modifying parameters in the `ase.conf` file located in the `config` directory. For more information, see Sideband ASE configuration or Inline ASE configuration
- Configuring deployment method: PingIntelligence supports on-premise deployment. For more information, see Configure deployment method.

Copyright © 2025 Ping Identity Corporation

ASE ports

ASE uses default ports as defined in the table below.

If any ports configured in the `ase.conf` file is unavailable, ASE will not start.

Port Number	Usage
80	Data port for HTTP and WebSocket connections. Accessible from any client (not secure). If you are installing ASE as a non-root user, choose a port that is greater than or equal to 1024.
443	Data port for HTTPS and Secure WebSocket (wss) connections. Accessible from any client. If you are installing ASE as a non-root user, choose a port that is greater than or equal to 1024.
8010	Management port used by CLI and REST API for managing ASE. Accessible from management systems and administrators.
8020	Cluster port used by ASE for cluster communication. Accessible from all cluster nodes.
8080	ABS ports used by ASE for outbound connections to ABS for sending access logs and receive client identifiers of suspected attacks.

Important

Management ports 8010 and 8020 should not be exposed to the Internet. If you are setting up the deployment in an AWS environment with security groups, use private IPs for ASE to ABS connections to avoid security group issues.

ASE deployment modes

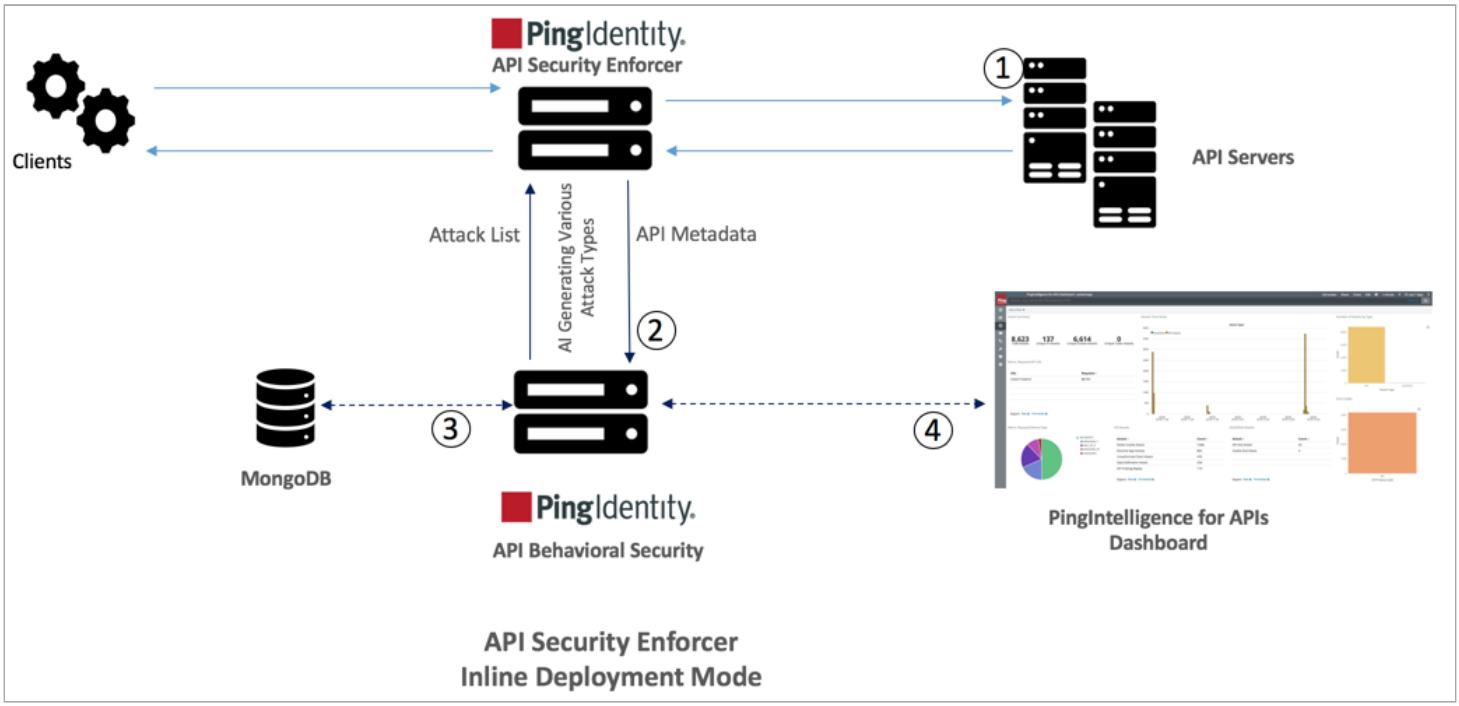
API Security Enforcer (ASE) supports REST and WebSocket APIs and can dynamically scale and secure system infrastructure.

ASE can be deployed in inline or sideband mode.

Inline mode

In the inline deployment mode, ASE sits at the edge of your network to receive the API traffic. It can also be deployed behind an existing load balancer, such as AWS ELB. In inline mode, ASE deployed at the edge of the datacenter terminates SSL connections from API clients. ASE then forwards the requests directly to the correct APIs and app servers, such as Node.js, WebLogic, Tomcat, PHP, etc.

Copyright © 2025 Ping Identity Corporation



To configure ASE to work in the inline mode, set `mode=inline` in the `/opt/pingidentity/ase/config/ase.conf` file.

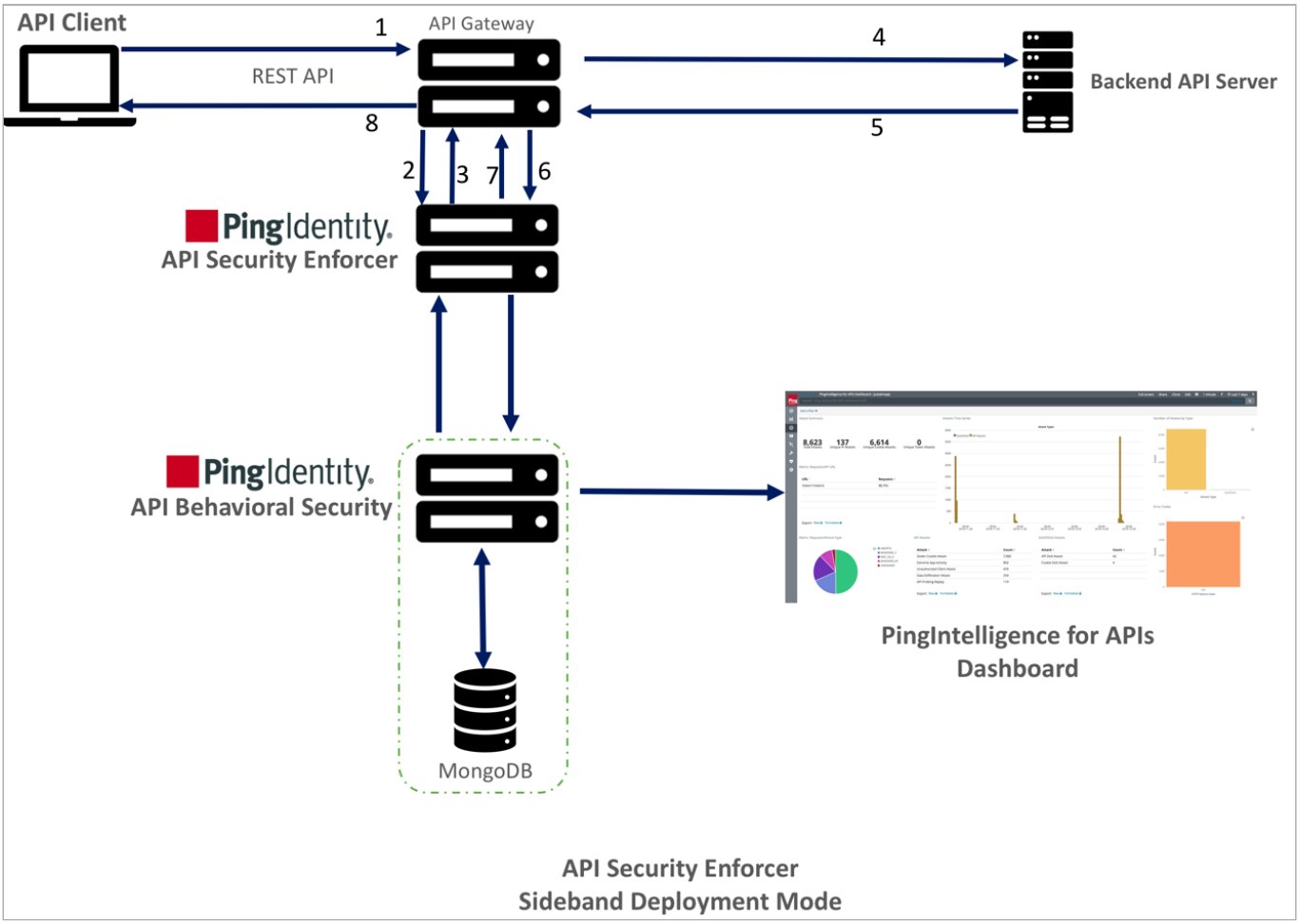
Some load balancers (for example, AWS ELB) require responses to keep alive messages from all devices receiving traffic. In an inline mode configuration, ASE should be configured to respond to these keep alive messages by updating the `enable_ase_health` variable in the `/opt/pingidentity/ase/config/ase.conf` file. When `enable_ase_health` is true, load balancers can perform an ASE health check using the following URL: `http(s)://<ASE Name>/ase` where *<ASE Name>* is the ASE domain name. ASE will respond to these health checks.

Sideband mode

When deployed in sideband mode, ASE works behind an existing API gateway. The API request and response data between the client and the backend resource or API server is sent to ASE. In this case, ASE does not directly terminate the client requests.

To configure ASE to work in inline mode, set `mode=sideband` in the `/opt/pingidentity/ase/config/ase.conf` file.

Copyright © 2025 Ping Identity Corporation



The following is a description of the traffic flow through the API gateway and Ping Identity ASE.

1. The API client sends a request to the API gateway.
2. The API gateway makes an API call to send the request detail in JSON format to ASE
3. ASE checks the request against a registered set of APIs and checks the origin IP against the AI-generated deny list. If all checks pass, ASE returns a `200-OK` response to the API gateway. Otherwise, a different response code is sent to the gateway. The request is also logged by ASE and sent to the AI engine for processing.
4. If the API gateway receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, the gateway returns a different response code to the client.
5. The response from the backend server is received by the API gateway.
6. The API gateway makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to the API gateway.
8. API gateway sends the response received from the backend server to the client.

> **Note**
>
> To complete the ASE sideband mode deployment, see the Integrate API gateways for sideband deployment.

Copyright © 2025 Ping Identity Corporation

Installing ASE

ASE supports RHEL 7.9 and Ubuntu 18.04 LTS. The provisioned infrastructure can be an EC2 instance, bare metal x86 server, and VMware ESXi.

About this task

Complete the following steps to install ASE. You can install ASE as a root user or as a non-root user. The example installation path assumes that you are root user. The installation works in a similar way for a non-root user.

Steps

1. Go to the Ping Identity Product Downloads site
2. Under PingIntelligence for APIs, click on View Now.
3. Choose the installation type and click Download.
4. After downloading the file, copy the ASE file to the `/opt` directory or any other directory where you want to install ASE.
5. Change the working directory.

   Choose from:

   - If you are installing as a root user, change the working directory to `/opt`
   - If you are installing as a non-root user, choose any other location.
6. At the command prompt, enter the following command to untar the ASE file:

   ```
   tar –zxvf  <filename>
   ```

   Example:

   ```
   tar –zxvf pi-api-ase-rhel-4.4.tar.gz
   ```
7. To verify that ASE successfully installed, enter the `ls` command at the command prompt. This should list the `pingidentity` directory and the build `.tar` file.

   Example:

   ```
   /opt/pingidentity/ase/bin/$ ls
   pingidentity pi-api-ase-rhel-4.4.tar.gz
   ```

Managing the ASE license

To start ASE, you need a valid PingIntelligence license.

About this task

Copyright © 2025 Ping Identity Corporation

There are two types of licenses:

- Trial license: Valid for 30 days. At the end of the trial period, ASE stops accepting traffic.
- Subscription license: Based on the subscription period. It is good practice to configure your email before configuring the license. ASE sends an email notification to the configured email ID in case the license has expired. Contact the Ping Identity Sales team for more information.

> **Note**
>
> If the subscription license has expired, ASE continues to run until a restart.

Steps

1. Configure the ASE license.
    1. If you have not already received a license, request a license file from the Ping Identity Sales team.

    > **Note**
    >
    > The name of the license file must be `PingIntelligence.lic` .

    2. Copy the license file to the `/opt/pingidentity/ase/config` directory and start ASE.
2. Update an existing license.
    1. If your existing license has expired, update the license by contacting the Ping Identity Sales team and replace the license file in the `/opt/pingidentity/ase/config` directory.

    > **Note**
    >
    > Make sure to stop and start ASE after the license file is updated.

Changing default settings

It is recommended that you change the default key and password in ASE.

About this task

Below is a list of commands to change the default values:

Changing `ase_master.key`

Create your own ASE master key to obfuscate keys and password in ASE.

Before you begin

ASE must be stopped before creating a new `ase_master.key` .

About this task

Copyright © 2025 Ping Identity Corporation

To create your own ASE master key:

Steps

1. Run the following command to create your own ASE master key to obfuscate keys and password in ASE: `generate_obfkey`.

```
/opt/pingidentity/ase/bin/cli.sh generate_obfkey -u admin -p admin
API Security Enforcer is running. Please stop ASE before generating new obfuscation master key
```

2. Stop ASE by running the following command:

```
/opt/pingidentity/ase/bin/stop.sh -u admin –p admin
checking API Security Enforcer status…sending stop request to ASE. please wait…
API Security Enforcer stopped
```

3. Enter the `generate_obfkey` command to change the default ASE master key:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin generate_obfkey
Please take a backup of config/ase_master.key, config/ase.conf,
config/abs.conf, config/cluster.conf before proceeding
Warning: Once you create a new obfuscation master key, you should
obfuscate all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/ase/config/ase_master.key already exist.
This command will delete it create a new key in the same file
Do you want to proceed [y/n]:
```

4. After a new ASE master key is generated, start ASE by entering the following command:

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 4.0...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Changing the key store password

You can change the key store password.

Before you begin

ASE must be running for updating the keystore password.

About this task

To change the key store password:

Copyright © 2025 Ping Identity Corporation

Steps

1. Enter the following command: `update_keystore_password`.

```
/opt/pingidentity/ase/bin/cli.sh update_keystore_password -u admin -p admin
New password >
New password again >
keystore password updated
```

> **Note**
>
> The default password is `asekeystore`.

Changing the admin password

You can change the default admin password.

About this task

To change the admin password:

Steps

1. Enter the following command:

```
/opt/pingidentity/ase/bin/cli.sh update_password -u admin -p
Old password >
New password >
New password again >
Password updated successfully
```

Obfuscating ASE keys and passwords

You must obfuscate the keys and passwords configured in `ase.conf`, `cluster.conf`, and `abs.conf` in the `config` directory.

About this task

ASE ships with a default `ase_master.key`, which is used to obfuscate the various keys and passwords. It is recommended to generate your own `ase_master.key`.

The following keys and passwords are obfuscated in the three configuration files:

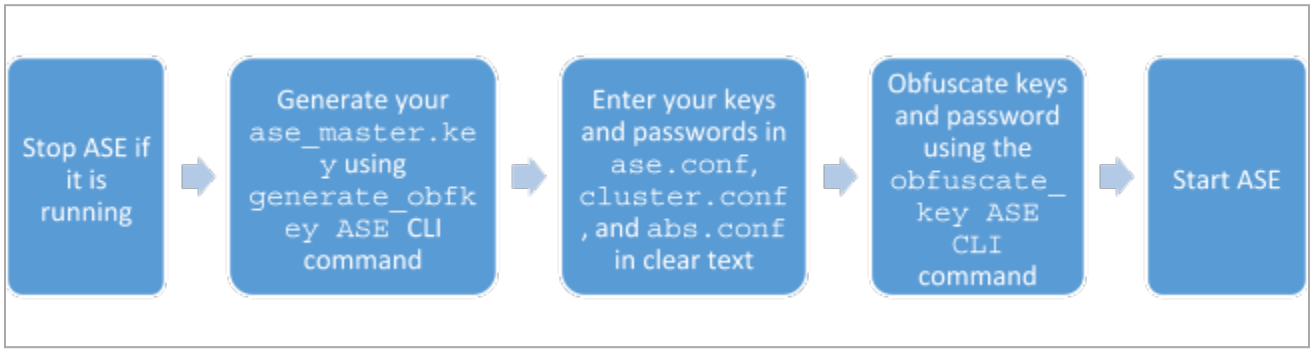
- `ase.conf`: Email and key store (PKCS#12) password
- `cluster.conf`: ABS access and secret key
- `abs.conf`: Cluster authentication key, `gateway_credential`

Copyright © 2025 Ping Identity Corporation

The new `ase_master.key` is used to obfuscate the keys and passwords in the various configuration files.

> **Note**
>
> During the process of obfuscating keys and passwords, ASE must be stopped.

The following diagram summarizes the obfuscation process:



Steps

1. To generate the `ase_master.key`, run the `generate_obfkey` command in the ASE command-line interface (CLI):

```
/opt/pingidentity/ase/bin/cli.sh generate_obfkey -u admin -p
Please take a backup of config/ase_master.key, config/ase.conf,
config/abs.conf, config/cluster.conf before proceeding

Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys

Warning: Obfuscation master key file /opt/pingidentity/ase/config/ase_master.key
already exist.

This command will delete it create a new key in the same file
Do you want to proceed [y/n]:y
creating new obfuscation master key
Success: created new obfuscation master key at
/opt/pingidentity/ase/config/ase_master.key
```

> **Important**
>
> In an ASE cluster, the new `ase_master.key` must be manually copied to each of the cluster nodes.

2. Enter the keys and passwords in clear text in `ase.conf`, `cluster.conf`, and `abs.conf`.
3. Run the `obfuscate_keys` command to obfuscate keys and passwords:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh obfuscate_keys -u admin -p
Please take a backup of config/ase_master.key, config/ase.conf, config/abs.conf, and config/
cluster.conf before proceeding
If config keys and password are already obfuscated using the current master key, it is not
obfuscated again
Following keys will be obfuscated:
config/ase.conf: sender_password, keystore_password
config/abs.conf: access_key, secret_key
config/cluster.conf: cluster_secret_key
Do you want to proceed [y/n]:y
obfuscating config/ase.conf, success
obfuscating config/abs.conf, success
obfuscating config/cluster.conf, success
```

4. Start ASE after keys and passwords are obfuscated.

Next steps

> **Important**
>
> After the keys and passwords are obfuscated, the `ase_master.key` must be moved to a secure location from ASE.

Tuning the host system for high performance

The API Security Enforcer (ASE) ships with a script to tune the host Linux operating system for handling high TCP concurrency and optimizing performance.

About this task

To understand the tuning parameters, see the tuning script comments. When running the tuning script, changes are displayed on the console to provide insight into system modifications.

To undo system changes, run the `untune` script.

> **Note**
>
> You should be a `root` user to run the **`tune`** and **`untune`** scripts.
> If you are installing ASE as a non-root user, run the **`tune`** script for your platform before starting ASE.

The following commands are for tuning RHEL. For tuning Ubuntu, use the Ubuntu tuning scripts.

Steps

- To tune the host system:

  1. In the command line, run the `tune_rhel7.sh` command.

     Example:

     ```
     /opt/pingidentity/ase/bin/tune_rhel7.sh
     ```

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> If ASE is deployed in a Docker Container, run the `tune` script on the host system, not in the container.

2. Close the current shell after running the `tune` script and proceeding to start ASE.

- To untune the host system and bring the system back to its original state, run the `untune_rhel7.sh` command.

  Example:

```
/opt/pingidentity/ase/bin/untune_rhel7.sh
```

Starting and stopping ASE

Learn how to start and stop ASE.

Before you begin

For ASE to start, the `ase_master.key` must be present in the `/opt/pingidentity/ase/config` directory. If you have moved the master key to a secured location for security reasons, copy it to the `config` directory before executing the `start` script.

Before starting ASE, make sure that the `nofile` limit in `/etc/security/limits.conf` is set to at least 65535 or higher on the host machine. Run the following command on the ASE host machine to check the `nofile` limit:

```
ulimit -n
```

About this task

You can start ASE in one of the following two ways:

- Using service script available in the `util` directory, or
- Using the `start.sh` script available in the `bin` directory.

Steps

1. Start ABS.

   Choose from:

   - To start ASE as a service:

     1. Navigate to the `util` directory and run the following command to install ASE as a service:

```
#sudo ./install-systemctl-service.sh pi-ase
```

     2. Start the service by entering the following command:

```
systemctl start pi-ase.service
```


Copyright © 2025 Ping Identity Corporation

- To start ASE using the `start.sh` script:

  1. Change the working directory to `bin`.
  2. Run the `start.sh` script:

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 5.0...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

2. Stop ASE.

Choose from:

- To stop ASE using the `stop.sh` script:

  1. Change the working directory to `bin`.
  2. Run the `stop.sh` script:

```
/opt/pingidentity/ase/bin/stop.sh -u admin –p admin
checking API Security Enforcer status…
sending stop request to ASE. please wait…
API Security Enforcer stopped
```

- To stop ASE using a service script:

  1. Run the following command:

```
systemctl stop pi-ase.service
```

Configuring SSL for client-side connection or external APIs

ASE supports both TLS 1.2 and SSLv3 for external APIs.

Before you begin

By default, SSLv3 is disabled due to security vulnerabilities. To change the default and enable SSLv3, stop ASE and then change `enable_sslv3` to true in `ase.conf` file. Restart ASE to activate SSLv3 protocol support. SSLv3 is only supported for client to ASE connections, not ASE to backend server connections.

```
; SSLv3
enable_sslv3=true
```

About this task

OpenSSL is bundled with ASE. The following are the version details:

- RHEL: OpenSSL 1.0.2k-fips 26 Jan 2017

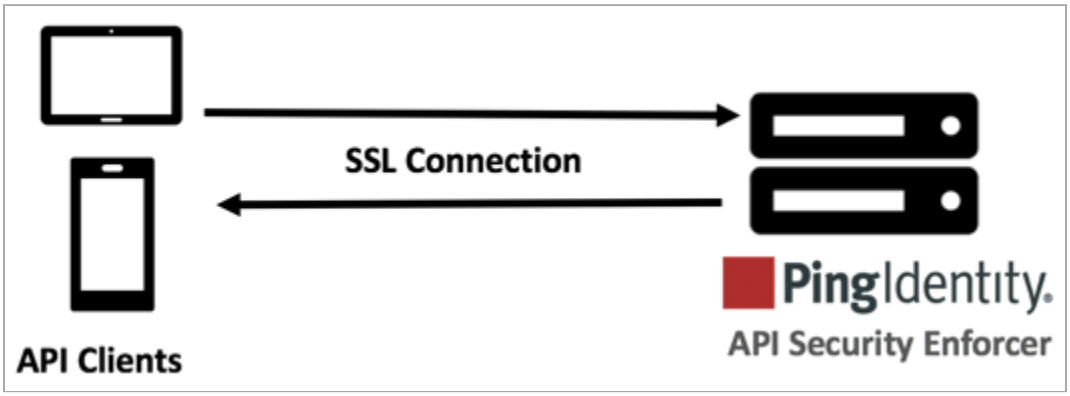
Copyright © 2025 Ping Identity Corporation

- Ubuntu: OpenSSL 1.0.2g 1 Mar 2016

You can configure SSL in ASE for client-side connection using one of the following methods:

- Using a CA-signed certificate
- Using a self-signed certificate
- Importing an existing certificate

The steps provided in this section are for the certificate and key generated for connections between the client and ASE as depicted in the diagram below:



In a cluster setup:

1. Stop all the ASE cluster nodes
2. Configure the certificate on the management node.

> **Note**
>
> For more information on management node, see API Security Enforcer Admin Guide.

3. Start the cluster nodes one by one for the certificates to synchronize across the nodes

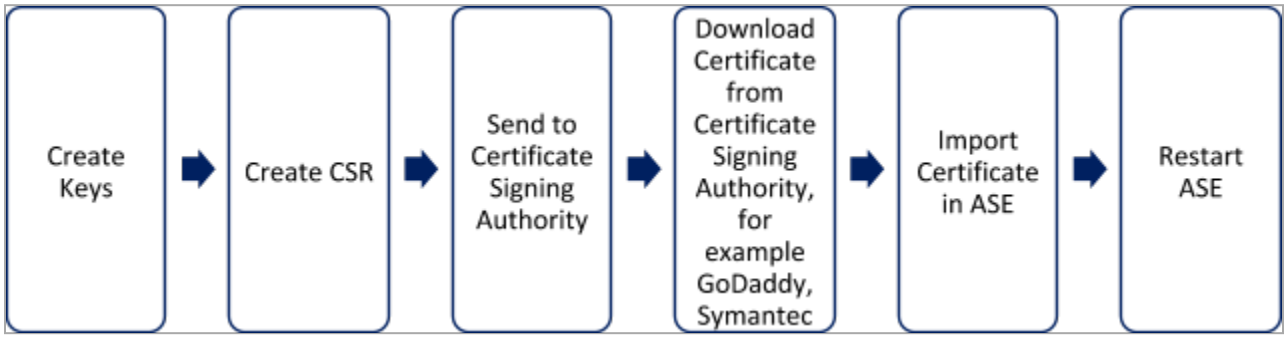
> **Important**
>
> You can also configure for Management APIs. For more information on configuring SSL for management APIs, see Configure SSL for Management APIs.

Using a CA-signed certificate

About this task

To use a Certificate Authority (CA)-signed SSL certificates, follow the process shown below to create a private key, generate a certificate signing request (CSR), and request a certificate:

Copyright © 2025 Ping Identity Corporation



Note

ASE internally validates the authenticity of the imported certificate.

Steps

1. Create a private key.

```
/optCDO:/content/authoring/nrc1651605112856.image/pingidentity/ase/bin/cli.sh create_key_pair -u
admin -p
Warning: create_key_pair will delete any existing key_pair, CSR and self-signed certificate
Do you want to proceed [y/n]:y
OK, creating new key pair. Creating DH parameter may take around 20 minutes. Please wait
Key created in keystore
dh param file created at /opt/pingidentity/ase/config/certs/dataplane/dh1024.pem
```

Note

ASE command-line interface (CLI) is used to create a 2048-bit private key and to store it in the key store.

2. Create a CSR.

```
/opt/pingidentity/ase/bin/cli.sh create_csr -u admin -p
Warning: create_csr will delete any existing CSR and self-signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >US
State > Colorado
Location >Denver
Organization >Pingidentity
Organization Unit >Pingintelligence
Common Name >ase
Generating CSR. Please wait...
OK, csr created at /opt/pingidentity/ase/config/certs/dataplane/ase.csr
```

Result:

ASE takes you through a CLI-based interactive session to create a CSR.

3. Upload the CSR created in step 2 to the CA-signing authority's website to get a CA-signed certificate.
4. Download the CA-signed certificate from the CA-signing authority's website.

Copyright © 2025 Ping Identity Corporation

5. Use the CLI to import the signed CA certificate into ASE. The certificate is imported into the key store.

```
/opt/pingidentity/ase/bin/cli.sh import_cert  <CA signed certificate path>  -u admin -p
Warning: import_cert will overwrite any existing signed certificate
Do you want to proceed [y/n]:y
Exporting certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

6. Restart ASE by first stopping and then starting ASE.

Using a self-signed certificate

About this task

A self-signed certificate is also supported for customer testing.

Steps

1. Create a private key: `/opt/pingidentity/ase/bin/cli.sh create_key_pair -u admin -p`

```
Warning: create_key_pair will delete any existing key_pair, CSR and self-signed certificate
Do you want to proceed [y/n]:y
OK, creating new key pair. Creating DH parameter may take around 20 minutes. Please wait
Key created in keystore
dh param file created at /opt/pingidentity/ase/config/certs/dataplane/dh1024.pem
```

> **Note**
>
> ASE CLI is used to generate a 2048-bit private key, which is in the `/opt/pingidentity/ase/config/certs/dataplane/dh1024.pem` directory.

2. Create a self-signed certificate. Use the CLI to produce a self-signed certificate located in `/pingidentity/ase/config/certs/dataplane/ase.csr`.

```
/opt/pingidentity/ase/bin/cli.sh create_self_sign_cert -u admin -p
Warning: create_self_sign_cert will delete any existing self-signed certificate
Do you want to proceed [y/n]:y
Creating new self-signed certificate
OK, self-sign certificate created in keystore
```

3. Restart ASE by stopping and starting.

Copyright © 2025 Ping Identity Corporation

Importing an existing certificate and key pair

About this task

To install an existing certificate, complete the following steps and import the certificate into ASE. If you have intermediate certificate from a CA, then append the content to your server `.crt` file.

Steps

1. Create the key from the existing `.pem` file:

```
openssl rsa -in private.pem -out private.key
```

2. Convert the existing `.pem` file to a `.crt` file:

```
openssl x509 -in server-cert.pem -out server-cert.crt
```

3. Import the key pair from step 2:

```
/opt/pingidentity/ase/bin/cli.sh import_key_pair private.key -u admin -p
Warning: import_key_pair will overwrite any existing certificates
Do you want to proceed [y/n]:y
Exporting key to API Security Enforcer...
OK, key pair added to keystore
```

4. Import the `.crt` file in ASE using the `import_cert` CLI command:

```
/opt/pingidentity/ase/bin/cli.sh import_cert server-crt.crt -u admin -p
Warning: import_cert will overwrite any existing signed certificate
Do you want to proceed [y/n]:y
Exporting certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

5. Restart ASE by stopping and starting.

Setting up an ASE cluster (optional)

For production environments, Ping Identity recommends setting up a cluster of ASE nodes for improved performance and availability.

Before you begin

Enable network time protocol (NTP) on each ASE node system. All cluster nodes must be in the same time zone.

About this task

Copyright © 2025 Ping Identity Corporation

To set up an ASE cluster node:

Steps

1. **Navigate to the `config` directory.**
2. **Edit the `ase.conf` file:**
    1. **Set `enable_cluster=true` for all cluster nodes.**
    2. **Confirm that the parameter `mode` is the same on each ASE cluster node, either `inline` or `sideband`.**

    > **Note**
    >
    > If parameter mode values do not match, the nodes will not form a cluster.

3. **Edit the `cluster.conf` file:**
    1. **Configure `cluster_id` with an identical value for all nodes in a single cluster (for example: `cluster_id=shopping`).**
    2. **Enter the port number in the `cluster_manager_port` parameter.**

    > **Note**
    >
    > ASE node uses this port number to communicate with other nodes in the cluster.

    3. **Enter an IPv4 address or hostname with the port number for `peer_node`, which is the first (or any existing) node in the cluster. Keep `peer_node` empty for the first cluster node.**
    4. **Provide the `cluster_secret_key`, which must be the same in each cluster node. It must be entered on each cluster node before the nodes to connect to each other.**

    Example:

    Below is a sample `cluster.conf` file:

Copyright © 2025 Ping Identity Corporation

```
; API Security Enforcer's cluster configuration.
; This file is in the standard .ini format. The comments start with a
; semicolon (;).
; Section is enclosed in []
; Following configurations are applicable only if cluster is enabled
; with true in ase.conf
; unique cluster id.
; valid character class is [ A-Z a-z 0-9 _ - . / ]
; nodes in same cluster should share same cluster id
cluster_id=ase_cluster

; cluster management port.
cluster_manager_port=8020

; cluster peer nodes.
; a comma-separated list of hostname:cluster_manager_port or
; IPv4_address:cluster_manager_port
; this node will try to connect all the nodes in this list
; they should share same cluster id
peer_node=

; cluster secret key.
; maximum length of secret key is 128 characters (deobfuscated length).
; every node should have same secret key to join same cluster.
; this field can not be empty.
; change default key for production.
cluster_secret_key=OBF:AES:nPJOh3wXQWK/BOHrtKu3G2SGiAEElOSvOFYEiWfIVSdu
```

4. After configuring an ASE node, start the node by running the following command:

```
/opt/pingidentity/ase/bin/start.sh
```

Scaling up the ASE cluster

Scale up the ASE cluster by adding nodes to an active cluster without disrupting traffic.

About this task

To add a new cluster node:

Steps

1. Enter the `peer_node` IP address or hostname in the `cluster.conf` file of the ASE node.
2. Start the ASE node.

   For more information, see Start the ASE node.

   Example:

Copyright © 2025 Ping Identity Corporation

If the IP of the first node is 192.168.20.121 with port 8020, then the `peer_node` parameter would be 192.168.20.121:8020.

```
; ASE cluster configuration. These configurations apply only when
; you have enabled cluster in the api_config file.
; Unique cluster ID for each cluster. All the nodes in the same cluster
; should have the same cluster ID.
cluster_id=ase_cluster
; Cluster management port.
cluster_manager_port=8020
; Cluster's active nodes. This can be a comma separated list of nodes in
; ipv4_address:cluster_manager_port format.
peer_node=192.168.20.121:8020
```

Result:

The new node will synchronize configuration and cookie data from the peer nodes. After loading, it will become part of the cluster.

Scaling down the ASE cluster

A node can be removed from an active cluster without disrupting traffic.

About this task

To remove a node from an active cluster:

Steps

1. Stop the ASE node to be removed.
2. Set the `enable_cluster` option as `false` in its `ase.conf` file.

   Result:

   The removed node retains the cookie and certificate data from when it was part of the cluster.

Deleting a cluster node

An inactive cluster node has either become unreachable or has been stopped.

About this task

When you delete a stopped cluster node, the operation does not remove cookie and other synchronized data.

Steps

1. To find which cluster nodes are inactive, use the `cluster_info` command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh cluster_info -u admin -p
cluster id : ase_cluster
cluster nodes
127.0.0.1:8020 active
Step 1.1.1.1:8020 active
Step 2.2.2.2:8020 inactive
172.17.0.4:8020(tasks.aseservice) active
172.17.0.5:8020(tasks.aseservice) inactive
tasks.aseservice2:8020 not resolved
```

2. Using the `cluster_info` command output, you can remove the inactive cluster nodes 2.2.2.2:8020 and 172.17.0.5:8020.
3. To delete the inactive node, use the `delete_cluster_node` command:

```
/opt/pingidentity/ase/bin/cli.sh delete_cluster_node  <IP:Port>
```

Stopping ASE cluster

You can stop an ASE cluster on any node in the cluster.

About this task

To stop the entire cluster:

Steps

1. Run the following command on any node in the cluster:

```
/opt/pingidentity/ase/bin/stop.sh cluster –u admin –p
```

Result:

When the cluster stops, each cluster node retains all the cookie and certificate data.

ASE and ABS integration

Integrate ASE and ABS.

The ABS engine installation process is summarized below:

- Connect ASE to the ABS AI engine for ASE to send access log files to ABS.
- Enable ASE to ABS engine communication: Just connecting ASE and the ABS engine does not mean that access logs are sent by ASE to ABS. ASE to ABS communication has to be enabled separately.

Copyright © 2025 Ping Identity Corporation

- Add API JSON files to ASE. The API JSON files define your API and its various parameters. For more information, see Defining an API using API JSON configuration file in inline mode.
- Train the ABS AI engine models to analyze and report on your API traffic.

Connecting ASE to the ABS AI engine

Connect ASE to the ABS AI engine.

Before you begin

The default ports for connection with ABS are 8080 and 9090. Run the `check_ports.sh` script on the ASE machine to determine accessibility of ABS. Input ABS host IP address and ports as arguments.

```
/opt/pingidentity/ase/util ./check_ports.sh {ABS IPv4:[port]}
```

About this task

To configure ASE for connecting with the ABS AI engine:

Steps

1. Update `abs.conf` located in the ASE `/opt/pingidentity/ase/config` directory with the ABS engine address and authentication keys:
    1. Configure `abs_endpoint` with the ABS engine management IP address/hostname and port number (default: 8080), which was configured in the `/opt/pingidentity/abs/config/abs.properties` file.

        > **Note**
        >
        > If ABS is in a different AWS security group, use a private IP address.

    2. Configure ABS `access_key` and `secret_key` using the key values from the `abs_init.js` file located in `/opt/pingidentity/abs/mongo`.

        Example:

        Below is a sample `abs.conf` file:

Copyright © 2025 Ping Identity Corporation

```
; API Security Enforcer ABS configuration.
; This file is in the standard .ini format. The comments start with a semicolon (;).
; Following configurations are applicable only if ABS is enabled with true.

; a comma-separated list of abs nodes having hostname:port or ipv4:port as an address.
abs_endpoint=127.0.0.1:8080

; access key for abs node
access_key=OBF:AES://ENOzsqOEhDBWLDY+pIoQ:jN6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0

; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw:zzi77GIFF5xkQJccjIrIVWU+RY5CxUhp3NLcNBel+3Q

; Setting this value to true will enable encrypted communication with ABS.
enable_ssl=true

; Configure the location of ABS's trusted CA certificates. If empty, ABS's certificate
; will not be verified
abs_ca_cert_path=
```

Important

Make sure that ASE and ABS are in the same time zone.

Enabling ASE to ABS engine communication

Enable communication between ASE and the ABS engine.

About this task

To start communication between ASE and the AI engine:

Steps

1. Run the following command:

```
./cli.sh enable_abs –u admin -p admin
```

2. To confirm an ASE node is communicating with ABS, issue the ASE status command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh status
API Security Enforcer
status              : started
mode                : inline
http/ws             : port 8080
https/wss           : port 8443
firewall            : enabled
abs                 : enabled, ssl: enabled (If ABS is enabled, then ASE is communicating with ABS)
abs attack          : disabled
audit               : enabled
ase detected attack : disabled
attack list memory  : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

Adding APIs to ASE

After the policy has been deployed to Apigee using the PingIntelligence automated policy tool, add APIs to ASE.

About this task

Refer to the following topics to define and add APIs to ASE:

Steps

- API naming guidelines
- Define and add an API JSON

Next steps

For more information on ASE sideband deployment, see Sideband API Security Enforcer.

Training the ABS AI engine

For ABS to start predicting various attacks types, the model needs to be trained.

About this task

The number of hours (default: 24 hours) is configurable for model training.

Steps

1. Set the value of `training_period` parameter using PingIntelligence Dashboard or the Global configuration update REST API.

   > **Note**
   >
   > For more detailed information about training the AI model, see Managing AI engine training.

2. Start the training.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> The training starts as soon as ABS receives the first API traffic from ASE and continues for the number of hours set in the `attack_initial_training` parameter. Training occurs automatically when a new API is added.

3. Verify training completion.

    1. To check the ABS training status, use the ABS admin API, which returns the training duration and prediction mode. The API URL for the admin API is `https://<ip>:<port>/v4/abs/admin`.

        > **Note**
        >
        > *ip* and *port* number are of the ABS machine.

        Result:

        The following is a snippet of the output of the admin API:

        ```
        "message": "training started at Thu Dec 26 12:32:59 IST 2019",
        "training_duration": "2 hours",
        "prediction": true
        ```

If the prediction variable is `true`, ABS has completed training and is discovering attacks. A `false` value means that ABS is still in training mode.

> **Note**
>
> ABS only detects attacks after the training period is over. During training, no attacks are generated.

API Publish Service

The API Publish Service publishes the changes made to the discovered APIs from the PingIntelligence Dashboard to the AI engine.

About this task

Complete the following steps to install the API Publish Service in your environment.

Installing the API Publish Service

Install the API Publish Service.

Before you begin

Before installing the API Publish Service:

- Install OpenJDK 11.0.2 on a 64-bit architecture machine. To verify the Java version, run the following command:

    ```
    # java -version
    ```

- Verify the supported operating systems.PingIntelligence supports RHEL 7.9 and Ubuntu 18.04 LTS.

Copyright © 2025 Ping Identity Corporation

About this task

You can install the API Publish Service as a root user or as a non-root user. The installation path in the steps below assumes that you are root user. The installation works in a similar way for a non-root user.

> **Note**
>
> The download site has a link to the consolidated build for ABS and API Publish. When extracting the tar in the `pingidentity` folder, there will be two folders:
>
> - `abs`
> - `apipublish`

Steps

1. **Go to the Ping Identity Product Downloads site.**
2. **Under PingIntelligence for APIs, click View Now.**
3. **Click Download under PingIntelligence for APIs Software.**
4. **Under Download AI Engine and Tools, click AI Engine 5.1.0.1.**
5. **After downloading:**

   Choose from:

   - **If you are installing as a root user, copy the build file to the `/opt` directory.**
   - **If you are installing as a non-root user, choose any other location.**
6. **At the command prompt, enter `# tar –zxvf <file_name>.`**

   Example:`# tar –zxvf pi-api-abs-5.1.tar.gz`

Default settings

The API Publish configuration file (`apipublish.properties`) is located in the `/pingidentity/apipublish/config/` directory. The following table explains the parameters and provides recommended values. You can change the default values based on your requirements.

Parameter	Description
`pi.apipublish.ssl.enabled-protocols`	**The supported SSL protocols. The default value is `TLSv1.2`.**
`pi.apipublish.ssl.ciphers`	**The supported `.ssl` ciphers. For the list of valid cipher names, see .oracle.com/en/java/javase/11/docs/specs/security/standard-names.html//[]. For multiple cipher names, use a comma to separate names in the list. For example: `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`, `TLS_DHE_RSA_WITH_AES_256_CBC_SHA256`.**

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`pi.apipublish.ssl.key-store`	The directory path of the key store. The default value is `config/ssl/apipublish.jks`.
`pi.apipublish.ssl.key-store-type`	The key store type. The default value is `JKS`.
`pi.apipublish.ssl.key-store-password`	The password of the JKS key store. PingIntelligence ships with a default obfuscated password. You can reset the password and obfuscate it.
`pi.apipublish.ssl.key-alias`	Alias for the SSL key. The default value is `pingidentity`.
`pi.apipublish.server.port`	Port for the API Publish Service and PingIntelligence Dashboard communication. The default value is `8050`.
`pi.apipublish.server.timezone`	Set the time zone to `utc` or `local`. The default timezone is `utc`.
`pi.apipublish.server.deployment_type`	The API Publish Service deployment mode. Valid values are `cloud` or `onprem`. The default value is `onprem`.
`pi.apipublish.datasource.data_dbname`	The MongoDB data database name. The default value is `abs_data`.
`pi.apipublish.datasource.metadata_dbname`	The MongoDB metadata database name.The default value is `abs_metadata`.
`pi.apipublish.datasource.mongo_rs`	Comma separated MongoDB replica set URI.
`pi.apipublish.datasource.mongo_ssl`	Set to `true` if MongoDB is configured to use SSL connections. The default value is `false`.
`pi.apipublish.datasource.mongo_auth_mechanism`	Defines the method in which MongoDB authenticates. The possible values are: • `NONE`: Set to `NONE` if authentication is not configured in MongoDB. • `DEFAULT`: Set to `DEFAULT` if you want to use a native MongoDB username and password. Provide the values in the next two variables. • `PLAIN`: Set to `PLAIN` if you want to use LDAP authentication. In this case, provide the LDAP username and password in the next two variables.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`pi.apipublish.datasource.mongo_certificate`	Set to true if you want to verify MongoDB SSL server certificate when the API Publish Service connects to MongoDB. The default value is `false`. **Note** Make sure `pi.apipublishservice.datasource.mongo_ssl` is set to `true` before setting `pi.apipublishservice.datasource.mongo_certificate` to `true`.
`pi.apipublish.datasource.username`	MongoDB username. The default value is `absuser`.
`pi.apipublish.datasource.password`	MongoDB password. The default value is `abs123`.

Obfuscating passwords

Using the command line interface (CLI), you can obfuscate the keys and passwords configured in `apipublish.properties`.

Before you begin
About this task

The API Publish Service is shipped with a default `apipublish_master.key`, which is used to obfuscate the various keys and passwords. It is recommended to generate your own `apipublish_master.key`. A default `jks_password` is configured in the `apipublish.properties` file.

The following keys and passwords are obfuscated:

- `mongo_password`
- `jks_password`

Note

During the process of obfuscation of keys and password, the API Publish Service must be stopped.

The following diagram summarizes the obfuscation process.

Steps

1. To generate the `apipublish_master.key`, run the `generate_obfkey` command in the CLI:

```
/pingidentity/apipublish/bin/cli.sh generate_obfkey -u admin -p admin
```


Copyright © 2025 Ping Identity Corporation

The new `apipublish_master.key` is used to obfuscate the passwords in `apipublish.properties` file.

2. Enter the keys and passwords in clear text in the `apipublish.properties` file.
3. Run the `obfuscate_keys` command to obfuscate keys and passwords:

```
/pingidentity/apipublish/bin/cli.sh obfuscate_keys -u admin -p admin
```

4. After the passwords are obfuscated, start the API Publish Service.

Next steps

> **Important**
>
> After the keys and passwords are obfuscated, the `apipublish_master.key` must be moved to a secure location.

Importing existing CA-signed certificates

Import existing CA-signed certificates.

Before you begin

To import the certificate authority (CA)-signed certificate, stop the API Publish Service if it is already running.

About this task

You can import your existing CA-signed certificate in the API Publish Service. Complete the following steps to import the CA-signed certificate.

> **Note**
>
> The API Publish Service is shipped with a default self-signed certificate with the Java key store at `/config/ssl/apipublish.jks`. The default password is set in the `apipublish.properties` file. The default password is obfuscated in the file. It is recommended to change the default passwords and obfuscate the new passwords. See Obfuscating passwords for steps to obfuscate passwords.

Steps

1. Export your CA-signed certificate to the PKCS12 store by entering the following command:

```
# openssl pkcs12 -export -in  <your_CA_cerficate.crt>  -inkey  <your_certificate_key.key>  -out
abs.p12 -name  <alias_name>
```

Example:

```
# openssl pkcs12 -export -in ping.crt -inkey ping.key -out abs.p12 -name exampleCAcertificate
Enter Export Password:
Verifying - Enter Export Password:
```

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> If you have an intermediate certificate from a CA, then append the content to the `<your_CA_certificate>.crt` file.

2. Import the certificate and key from the PKCS12 store to the Java key store by entering the command below.

```
# keytool -importkeystore -destkeystore apipublish.jks -srckeystore abs.p12 -srcstoretype PKCS12 -
alias <alias_name> -storetype jks
```

The command requires the destination key store password. The destination key store password entered in the command should be same that is configured in the `apipublish.properties` file.

The following is a snippet of the `apipublish.properties` file where the destination key store password is stored. The password is obfuscated.

```
# Java Keystore password
jks_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCulhAUKj7ww4=
```

Example:

```
# keytool -importkeystore -destkeystore apipublish.jks -srckeystore abs.p12 -srcstoretype PKCS12 -
alias exampleCAcertificate -storetype jks
Importing keystore apipublish.p12 to abs.jks...
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
```

3. Copy the `apipublish.jks` file created in step 2 to `/config/ssl` directory.

4. Start the API Publish Service by running the following command:

```
# ./bin/start.sh
```

Starting and stopping the API Publish Service

Start and stop the API Publish Service.

Before you begin

For the API Publish Service to start, the `apipublish_master.key` must be present in the `apipublish/config` directory. If you have moved the master key to a secured location for security reasons, copy it to the `config` directory before starting the service.

About this task

You can start the API Publish Service in one of the following two ways:

- Using a service script available in the `bin` directory
- Using the `start.sh` script available in the `bin` directory

Copyright © 2025 Ping Identity Corporation

Steps

1. Start API Publish.

   Choose from:

   - To start API Publish as a service:

     1. Navigate to the `bin` directory and run the following command to install API Publish as a service:

        ```
        #sudo ./install-systemctl-service.sh pi-apipublish
        ```

     2. Start the service by entering the following command:

        ```
        systemctl start pi-apipublish.service
        ```

   - To start API Publish using the `start.sh` script:

     1. Run the `start.sh` script located in the `/pingidentity/apipublish/bin` directory:

        ```
        $ ../bin/start.sh
        ```

2. Stop API Publish.

   Choose from:

   - To stop API Publish using a service script:

     1. Run the following command to stop the API Publish Service:

        ```
        systemctl stop pi-apipublish.service
        ```

   - To stop API Publish using the `stop.sh` script:

     1. Run the `stop.sh` script available in the `bin` directory:

        ```
        # ../bin/stop.sh
        ```

Installing the PingIntelligence Dashboard

Learn how to install the PingIntelligence Dashboard.

Before you begin

Ensure that the following prerequisites are met:

- Server: 8 core CPU, 16 GB, 1 TB HDD

Copyright © 2025 Ping Identity Corporation

- Operating system: RHEL 7.9 or Ubuntu 18.04 LTS
- OpenJDK: 11.0.2
- SSL certificate: One private key and certificate. By default, PingIntelligence Dashboard uses the private key and certificate shipped with the binary.
- Password: If you want to change the default password, set a minimum 8 character password.
- API Behavioral Security (ABS): ABS URL, access, and secret key. Make sure that ABS is reachable from the PingIntelligence Dashboard machine.
- API Security Enforcer (ASE): ASE management URL, access, and secret key. Make sure that ASE is reachable from the PingIntelligence Dashboard machine.

> **Note**
>
> Connecting the Dashboard to ASE is optional. Functionality, such as adding discovered APIs to ASE and attack management, will be limited.

Make sure the following default port numbers are available:

- PingIntelligence Dashboard (WebGUI) server: 8030. Port number 8030 should be exposed to public internet. Make sure that your organization's firewall allows access to this port.
- Elasticsearch: 9200
- Dataengine: 8040
- H2 database: 9092. The H2 database is installed and runs as a part of the PingIntelligence Dashboard.

Make sure you have one of the following supported browsers installed. The following table shows the compatibility of PingIntelligence for APIs Dashboard with different browsers and their versions.

Operating System	Google Chrome	Mozilla Firefox	Apple Safari	Microsoft Edge
Mac OS Mojave -10.14	Version 56.0 and later	Version 69.0 and later	Version 12.0 and later	
Mac OS Sierra -10.12	Version 56.0 and later	Version 69.0 and later	Version 10.1 and later	
Mac OS High Sierra -10.13	Version 56.0 and later	Version 69.0 and later	Version 11.1 and later	
Mac OS Catalina -10.15	Version 56.0 and later	Version 69.0 and later	Version 13.0 and later	
Windows 8.1	Version56.0 and later	Version 69.0 and later		
Windows 10	Version 56.0 and later	Version 69.0 and later		Version 79.0 and later

Ensure you have completed the following configuration for the operating system:

- Increase the `ulimit` to 65536:

Copyright © 2025 Ping Identity Corporation

```
# sudo sysctl -w fs.file-max=65536
# sudo sysctl -p
```

- Increase the `vm.max_map_count` limit to 262144:

```
# sudo echo "vm.max_map_count=262144" >> /etc/sysctl.conf
# sudo sysctl -p
```

- Set `JAVA_HOME` to the *<jdk_install>* directory and add `<jdk_install>/bin` to the system `PATH` variable. *<jdk_install_dir>* is the directory where JDK is installed.
- Choose the *<pi_install_dir>* directory. The *<pi_install_dir>* directory is the directory where the PingIntelligence Dashboard is installed. This directory should be readable and writable by the logged in user.

About this task

Installing the PingIntelligence for APIs Dashboard automatically installs Elasticsearch.

There are two preconfigured login users in PingIntelligence Dashboard:

- `admin`
- `ping_user`

Multiple `admin` and `ping_user` can simultaneously sign on to PingIntelligence Dashboard. The admin user has full access toPingIntelligence Dashboard. An `admin` can view the dashboard of various APIs as well as tune threshold and unblock a client identifier. `ping_user` can only view the API dashboard. A total of 25 `admin` and `ping_user` can sign on simultaneously.

To install the PingIntelligence Dashboard:

Steps

1. Create a *<ping_install_dir>* directory on your host machine.

   Make sure that the user has read and write permissions for the *<ping_install_dir>* directory.

2. Download the PingIntelligence Dashboard binary.

   1. Under Download AI Engine and Tools, click Dashboard 5.1.0.1.

3. Download Elasticsearch 6.8.1 (macOS/RHEL).
4. Change the directory to *ping_install_dir*:

```
# cd pi_install_dir
```

5. Untar the PingIntelligence Dashboard:

```
# tar -zxf pi-api-dashboard-5.1.tar.gz
```

6. Add the MongoDB IP address in `webgui.properties`.

Copyright © 2025 Ping Identity Corporation

7. Copy the MongoDB certificate to the Dashboard virtual machine (VM).
8. Import the MongoDB certificate to `webgui.jks` using the following command:

```
keytool -import -keystore dataengine.jks -storetype JKS -storepass changeme -alias mongo -file
mongo.crt -noprompt
```

9. Change the directory to `pingidentity/webgui/`:

```
# cd pingidentity/webgui/
```

10. Install the PingIntelligence Dashboard by entering the following command and follow the instructions displayed on the prompt:

```
# ./bin/pi-install-ui.sh
```


Copyright © 2025 Ping Identity Corporation

```
# ./bin/pi-install-ui.sh

elasticsearch-7.13.4.tar.gz file path >
Use bundled ssl key and self signed certificate for ui server [y/n]?  >[y]
Use default password [changeme] for all components and users [y/n]?  >[y]
ABS url  >[https://127.0.0.1:8080]
ABS access key  >[abs_ak]
ABS secret key  >[abs_sk]
API Service URL  >[https://127.0.0.1:8050]
Kafka Host:Port >[127.0.0.1:9093]
Kafka Authentication username  >[pi4api_de_user]
Kafka Group ID  >[pi4api.data-engine]
ASE management url  >[]
extracting elasticsearch package
creating elasticsearch config keystore
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
Created elasticsearch keystore in <pi_install_dir>/pingidentity/elasticsearch/config/
elasticsearch.keystore
elasticsearch config keystore created
Generating a 2048 bit RSA private key
.........................................+++
.....................+++
writing new private key to 'config/ssl/autogen_es.key'
-----
creating password protected pkcs#12 keystore for elasticsearch private key and certificate
pkcs#12 keystore created at config/ssl/elastic-certificates.p12
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
configuring elasticsearch. Please wait 15 seconds
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will
likely be removed in a future release.
elasticsearch config is completed
configuring dataengine
configuring webgui
starting webgui for configuration update
WebGUI configured for UTC timezone.
WebGUI 5.1 starting...
please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for more details
success: password updated.
Note: All active sessions for this user are invalidated. Login with new credentials
success: password updated.
Note: All active sessions for this user are invalidated. Login with new credentials
WebGUI 5.1
WebGUI is stopped.
webgui configuration done

UI configuration done
writing internal credentials to <pi_install_dir>/pingidentity/webgui/install/webgui_internal.creds
Start UI [y/n]?  >[y]
starting elasticsearch...
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
```

Copyright © 2025 Ping Identity Corporation

```
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will
likely be removed in a future release.
elasticsearch started
starting dataengine
Data Engine configured for UTC timezone.
PingIntelligence Data Engine 5.1 starting...
Data-Engine started
starting webgui
WebGUI configured for UTC timezone.
WebGUI 5.1 starting...
please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for more details
Please access WebGUI at https://<pi_install_host>:8030

<pi_install_host> can be ip address, hostname or fully qualified domain name of this server.
<pi_install_host> should be reachable from your computer.

Credentials:
  1) Username: admin
     Password: changeme
  2) Username: ping_user
     Password: changeme

Important Actions:
1) Credentials for all internal components are available in <pi_install_dir>/pingidentity/webgui/
install/webgui_internal.creds file.
    Move this file from this server and securely keep it elsewhere.
    For any debugging purposes you will be asked to get credentials for a component from this file.
2) Following obfuscation master keys are auto-generated
      <pi_install_dir>/pingidentity/webgui/config/webgui_master.key
      <pi_install_dir>/pingidentity/dataengine/config/dataengine_master.key
```

Important

The `ASE management url` is an optional parameter.

11. Verify installation by checking the process IDs (`pid`) of each component in the following locations:
    1. Elasticsearch: `<pi_install_dir>/elasticsearch/logs/elasticsearch.pid`
    2. Dataengine: `<pi_install_dir>/dataengine/logs/dashboard.pid`
    3. Web GUI: `<pi_install_dir>/webgui/logs/webgui.pid`
12. Tune the Dashboard performance parameters by configuring the following three parameters for better performance.

Note

Note the following tuning parameters if you have your setup of Elasticsearch. If you have used PingIntelligence automated deployment or the `pi-install-ui.sh` script to deploy the Dashboard, the tuning of the parameters below is done as part of installation.

Copyright © 2025 Ping Identity Corporation

Parameter	Description	Location
Elasticsearch		
`-Xms` and `-Xmx`	◦ `-Xms` : Defines the minimum heap size of Elasticsearch. Set to 4 GB as `Xms4g` . ◦ `-Xmx` : Defines the maximum heap size of Elasticsearch. Set to 4 GB as `Xmx4g` .	`$ES_HOME/config/jvm.options`
`thread_pool.search.size`	Defines thread pool size for count/ search/suggest operations in Elasticsearch. Configure to 50% of total CPUs allocated.	`$ES_HOME/config/ elasticsearch.yml`

Troubleshooting:

To detect and mitigate attacks such as cross-site scripting (XSS), the PingIntelligence Dashboard implements Content Security Policy (CSP). The following are the configuration details:

```
Response header - Content-Security-Policy
```

```
Response header value - default-src 'self'; font-src 'self' use.typekit.net; script-src 'self'
use.typekit.net; style-src 'self' 'unsafe-inline' use.typekit.net p.typekit.net; img-src 'self'
data: p.typekit.net;
```

Starting and stopping the PingIntelligence Dashboard

Start and stop the PingIntelligence Dashboard.

About this task

You can choose to start and stop all the components together or individually.

Steps

1. To start and stop components:

   Choose from:

   ◦ Start and stop components together.

   ◦ It is recommended to start and stop components together using the following command:

Copyright © 2025 Ping Identity Corporation

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/start-all.sh

starting elasticsearch...
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and
will likely be removed in a future release.
elasticsearch started
starting data-engine
Data Engine configured for UTC timezone.
PingIntelligence Data Engine 5.1 starting...
data-engine started
starting webgui
WebGUI configured for UTC timezone.
WebGUI 5.1 starting...
please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for more details
success: all ui components started
```

- To stop all components together, enter the following command:

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/stop-all.sh

WebGUI 5.1
WebGUI is stopped.
PingIntelligence Data Engine 5.1
PingIntelligence Data Engine is stopped.
elasticsearch stopped
success: all ui components stopped
```

- Start and stop components individually.

    1. Start components in the following order:

        1. Start Elasticsearch by entering the following command:

```
# cd  <pi_install_dir>/pingidentity/elasticsearch
# ./bin/elasticsearch -d -p logs/elasticsearch.pid
```

        If Elasticsearch is running as a service, use the following command:

```
# sudo systemctl start pi-elasticsearch.service
```

        2. Start the Dashboard by entering the following the command:

Copyright © 2025 Ping Identity Corporation

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/start.sh

WebGUI configured for UTC timezone.
WebGUI 5.1 starting...
please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for
more details
```

If the Dashboard is running as a service, use the following command:

```
# sudo systemctl start pi-dashboard.service
```

3. Start the Web GUI by entering the following command:

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/start.sh
```

If the Web GUI is running as a service, use the following command:

```
# sudo systemctl start pi-webgui.service
```

2. Stop components in the following order:

1. Stop the Web GUI by entering the following command:

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/stop.sh
```

If Web GUI is running as a service, use the following command:

```
# sudo systemctl stop pi-webgui.service
```

2. Stop the Dashboard by entering the following command:

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/stop.sh

WebGUI 5.1
WebGUI is stopped.
```

If the Dashboard is running as a service, use the following command:

```
# sudo systemctl stop pi-dashboard.service
```

Copyright © 2025 Ping Identity Corporation

3. Stop Elasticsearch by entering the following command:

```
# cd  <pi_install_dir>/pingidentity/elasticsearch
# kill -15 "$(<logs/elasticsearch.pid)"
```

If Elasticsearch is running as a service, use the following command:

```
# sudo systemctl stop pi-elasticsearch.service
```

ABS reporting

The ABS AI engine generates attack, metric, and forensics reports, which are accessed using the ABS REST API to access JSON formatted reports.

Ping Identity provides Postman collections to generate various API reports. You can use any other tool to access the reports using the URLs documented in the ABS Admin Guide.

Install Postman with PingIntelligence for APIs Reports

Ping Identity provides configuration files that are used by Postman to access the ABS REST API JSON information reports. Make sure to install Postman 6.2.5 or higher.

Using ABS self-signed certificates with Postman

API Behavioral Security (ABS) ships with a self-signed certificate.

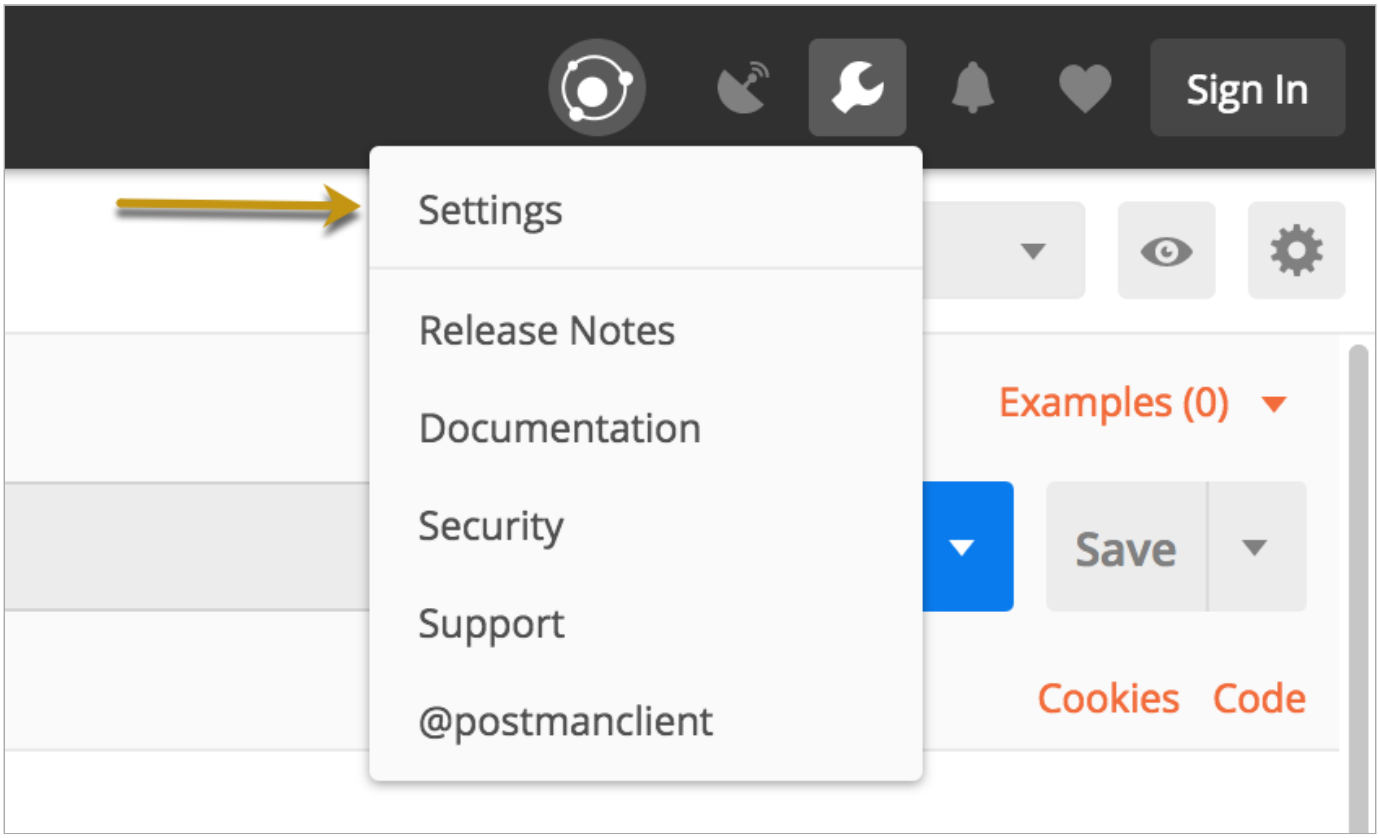
About this task

If you want to use Postman with the self-signed certificate of ABS, you can disable the certificate verification option from Postman's settings.

Steps

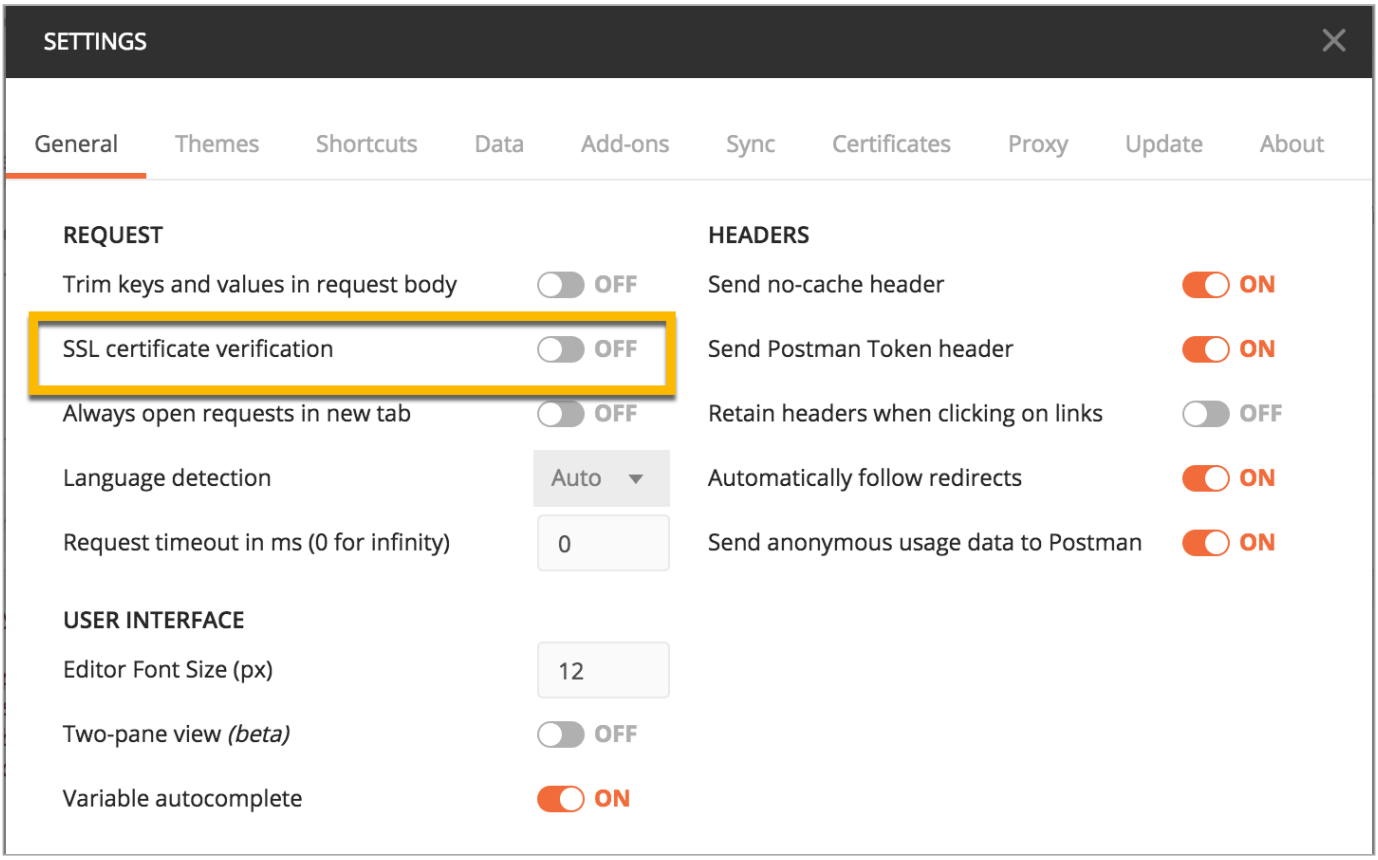
1. Click on the Wrench  icon on the top-right corner of Postman client. A drop-down window is displayed.
2. Select Settings from the drop-down menu:

Copyright © 2025 Ping Identity Corporation



3. **In the Settings window, click the toggle to turn off certificate verification by clicking on the SSL certificate verification button:**

Copyright © 2025 Ping Identity Corporation



Viewing ABS reports in Postman

View the API Behavioral Security (ABS) reports in Postman.

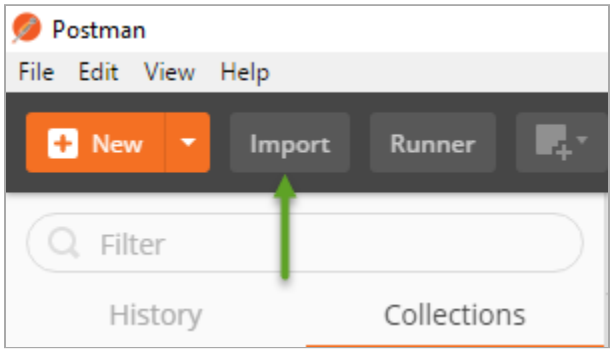
About this task

To view the ABS reports, complete the following steps:

Steps

1. Download the ABS environment and ABS reports JSON files from the `API Reports Using Postman` folder on Ping Identity download site. These configuration files will be used by Postman.
2. Download and install the Postman application 6.2.5 or higher.
3. In Postman, import the two Ping Identity files downloaded in step 1 by clicking the Import button.

Copyright © 2025 Ping Identity Corporation



4. After importing the files, click the ⚙ icon in the upper-right corner.
5. In the Manage Environments pop-up window, click ABS_4.3_Environment.
6. In the pop-up window, configure the following values and then click Update:
    - Server: IP address of the ABS node for which the `dashboard_node` was set to `true` in the `abs.properties` file.
    - Port: Port number of the ABS node.
    - Access_Key_Header and Secret_Key_Header: Use the admin user or restricted user header. A restricted user sees obfuscated value of OAuth token, cookie, and API keys. For more information of different types of user, see ABS users for API reports.
    - Access_Key and Secret_Key: The access key and secret key configured in `opt/pingidentity/mongo/abs_init.js` for either an admin or restricted user. Make sure that access key and secret key corresponds to the admin or restricted user header configured.
    - API_Name: The name of the API for which you want to generate the reports.
    - Later_Date: A date that is more recent in time. For example, if the query range is between March 12 and March 14, then the later date would be March 14.
    - Earlier_Date: A date that is past in time. For example, if the query range is between March 12 and March 14, then the earlier date would be March 12.

    > **Note**
    > Do not edit any fields that start with the word `System`.
7. In the main Postman window, select the report to display on the left column and then click Send. The ABS external REST APIs section provides detailed information on each API call and the JSON report response.

Integrate API gateways for sideband deployment

If you have deployed ASE in the sideband mode, the next step is to integrate your API gateway with PingIntelligence products.

To deploy ASE in the sideband mode, set `mode=sideband` in the `/opt/pingidentity/ase/config/ase.conf` file. This is the only configuration required on ASE for sideband deployment. For more information on ASE in sideband, see Sideband API Security Enforcer.

Copyright © 2025 Ping Identity Corporation

After you have completed installation and integration of ASE and ABS deployment, integrate one of the following API gateways with PingIntelligence components and start sending the API traffic to your API gateway:

- Akana API gateway sideband integration
- Apigee integration
- AWS API Gateway integration
- Azure APIM sideband integration
- Axway sideband integration
- CA API gateway sideband integration
- F5 BIG-IP integration
- IBM DataPower Gateway sideband integration
- Kong API gateway integration
- MuleSoft sideband integration
- NGINX sideband integration
- NGINX Plus sideband integration
- PingAccess sideband integration
- WSO2 integration

Installing the PingIntelligence machine learning service

Install the PingIntelligence machine learning (ML) service node.

Before you begin

- Install Python 3.8.13 in the machine learning service node.
- Install pip3 and ensure it points to Python 3.8.13.
- Install Python as the root user and make sure Python3 is pointing to python 3.8.13 (for example, by creating a soft link: `ln -s /usr/local/bin/python3.8 /usr/bin/python3`).
- Make sure management port 8090 is available.

About this task

To install the PingIntelligence machine learning service:

Steps

1. Go to the Ping Identity Product Downloads website.
2. Under PingIntelligence for APIs, click View Now.
3. Click Download under PingIntelligence for APIs.

Copyright © 2025 Ping Identity Corporation

4. Select a deployment method.
5. After downloading the file, copy the machine learning (ML) service file to the `/opt` directory or any other directory where you want to install the ML service.
6. Change the working directory:

   Choose from:

   - If you are installing as a root user, change the working directory to `/opt`.
   - If you want to install as a non-root user, choose any other location.
7. At the command prompt, type the following command to untar the ML service build file:

   ```
   tar –zxvf  <filename>
   ```

   Example:

   ```
   tar –zxvf pi-api-mlservice-5.2.tar.gz
   ```
8. To verify that the ML service successfully installed, type the `ls` command at the command prompt.

   Example:

   ```
   /opt/pingidentity/$ ls pingidentity
   Ml_service
   ```

   Result:

   This command lists the `pingidentity` directory and `ml_service` directory.
9. Install the requirements:

   ```
   pip3 install -r /opt/pingidentity/ml_service/lib/requirements.txt
   ```
10. To update the `properties.rc` file:
    1. Add Kafka, ABS, and MongoDB details to the `properties.rc` file in `config` directory.
    2. Copy the MongoDB (`mongo.crt`) and Kafka (`kafka.crt`) certificates to the `ml_service` node and provide that path in the `properties.rc` file.
    3. Set `export mongo_ssl` to `true`.

       ```
       export mongo_ssl="true"
       ```
    4. Set `export kafka_producer_insync_replicas` to 1.

       ```
       export kafka_producer_insync_replicas=1
       ```

Copyright © 2025 Ping Identity Corporation

Next steps

- Changing default settings
- Obfuscating keys and passwords
- Starting and stopping the server

Changing default settings

Change the default settings of the machine learning (ML) service to obfuscate keys and passwords.

Before you begin

Before creating a new `ml service` master key, stop the ML service with the following command:

```
/opt/pingidentity/ml_service/bin/stop.sh
```

About this task

To change the default settings:

Steps

1. To create your own `ml service` master key to obfuscate keys and passwords in the ML service, run the `generate_obfkey` command.

   ```
   /opt/pingidentity/ml_service/bin/cli.sh generate_obfkey
   ```

   Result:

   A master key is generated and written to the `/opt/pingidentity/ml_service/config/ml_master.key` file.

Obfuscating keys and passwords

Obfuscate keys and passwords in the `properties.rc` file.

About this task

To obfuscate keys and passwords configured in the `properties.rc` file:

Steps

1. Add the actual keys in the `properties.rc` file.
2. To obfuscate keys and passwords, run the following command:

   ```
   /opt/pingidentity/ml_service/bin/cli.sh obfuscate_keys
   ```

   Result:

Copyright © 2025 Ping Identity Corporation

The following keys are obfuscated in the configuration files:

- `Kafka_producer_jaas_password`
- `Kafka_consumer_password`
- `Mongo_password`

Starting and stopping the server

Start and stop the server when installing the PingIntelligence machine learning service.

About this task

To start and stop the server:

Steps

- Start the server using the `start.sh` script:

```
./bin/start.sh
Starting ML Service
ML Service started successfully
```

> **Note**
>
> You can view the logs in `/opt/pingidentity/ml_service/logs/ml_service.log`.

- Stop the server using the `stop.sh` script:

```
./bin/stop.sh
```

PingIntelligence Kubernetes deployment

Install PingIntelligence for APIs in the Kubernetes cluster, Amazon Elastic Kubernetes Service (EKS).

PingIntelligence ships with Helm-Chart that is packaged with the Docker toolkit and can be used to deploy PingIntelligence in a Kubernetes cluster.

PingIntelligence creates the following resources in the Kubernetes cluster:

- Seven statefulsets with one container each for:
  - MongoDB
  - API Behavioral Security (ABS) AI engine
  - API Security Enforcer (ASE)
  - API Publish

Copyright © 2025 Ping Identity Corporation

- PingIntelligence Dashboard
- Apache Zookeeper
- Kafka

• Six external services (LoadBalancer type), one each for (Configurable to expose):

- MongoDB
- ABS AI engine
- ASE
- API Publish
- Zookeeper
- Kafka

> **Note**
>
> Each component has an external service of type LoadBalancer that can be exposed by setting the flag in `values.yaml` (`expose_external_service: false`). By default, this value is true for ASE. The Dashboard will always be exposed since it must be accessible externally.

• Six internal services (clusterIP type), one each for:

- MongoDB
- ABS AI engine
- ASE
- API Publish
- Zookeeper
- Kafka

PingIntelligence Kubernetes supports RHEL 7.9.

> **Note**
>
> This deployment of PingIntelligence on a Kubernetes cluster node is suitable for Amazon EKS.

The Kubernetes cluster can be configured on the Amazon EKS. You can install PingIntelligence on a Kubernetes cluster node using Amazon EKS.

Copyright © 2025 Ping Identity Corporation

Amazon EKS

Deploying PingIntelligence using Amazon EKS
About this task

To deploy PingIntelligence on a Kubernetes cluster node using Amazon EKS:

Steps

1. **Create an Amazon EKS cluster on a RHEL or host.**

   You can use either `eksctl` or AWS command-line interface (CLI) for creating the Kubernetes cluster. Refer to the following links when creating and managing the Kubernetes cluster using Amazon EKS.

   For more information, see the following:

   - Getting started with Amazon EKS – eksctl
   - Getting started with Amazon EKS – AWS CLI

2. **Follow the steps in Deploying PingIntelligence in Kubernetes cluster and deploy PingIntelligence on the Kubernetes cluster that you created in step 1.**

Copyright © 2025 Ping Identity Corporation

Kubernetes cluster

Deploying PingIntelligence in Kubernetes cluster
Before you begin

Make sure you have a valid PingIntelligence license.

About this task

The Helm-Chart to deploy PingIntelligence in Kubernetes is shipped inside the Docker toolkit.

To deploy PingIntelligence in a Kubernetes cluster:

Steps

1. Download the PingIntelligence Docker toolkit from the PingIntelligence software download site.
2. Untar the Docker toolkit by running the following command:

```
tar -zxf <PingIntelligence Docker toolkit>
```

Directory structure:

```
pingidentity/
|-- docker-toolkit
`-- helm-chart
```


Copyright © 2025 Ping Identity Corporation

Directory structure for Helm-Chart:

```
helm-chart/
`-- pi4api
    |-- Chart.yaml
    |-- templates
    |   |-- abs_deployment.yaml
    |   |-- abs_external_service.yaml
    |   |-- abs_internal_service.yaml
    |   |-- apipublish_deployment.yaml
    |   |-- apipublish_external_service.yaml
    |   |-- apipublish_internal_service.yaml
    |   |-- ase_deployment.yaml
    |   |-- ase_external_service.yaml
    |   |-- ase_internal_service.yaml
    |   |-- dashboard_deployment.yaml
    |   |-- dashboard_external_service.yaml
    |   |-- _helpers.tpl
    |   |-- kafka_deployment.yaml
    |   |-- kafka_external_service.yaml
    |   |-- kafka_internal_service.yaml
    |   |-- license_configmap.yaml
    |   |-- mongo_deployment.yaml
    |   |-- mongo_external_service.yaml
    |   |-- mongo_internal_service.yaml
    |   |-- namespace.yaml
    |   |-- service_account.yaml
    |   |-- zookeeper_deployment.yaml
    |   |-- zookeeper_external_service.yaml
    |   `-- zookeeper_internal_service.yaml
    |-- values.yaml
    `-- version.txt
```

`pi4api`	Folder contains all PingIntelligence resources to deploy on Kubernetes
`Chart.yaml`	PingIntelligence chart details
`templates`	PingIntelligence deployable resources `yaml` files
`values.yaml`	PingIntelligence configuration example file

3. Build the PingIntelligence Docker images by completing the steps in Building the PingIntelligence Docker images.

4. Create a custom `values.yaml` file with all the configurations for PingIntelligence.

> **Note**
>
> The `values.yaml` packed with Helm-Chart is for example purposes and with default values. Override the values in a custom `values.yaml`, or change the values in the default `.yaml`.

Copyright © 2025 Ping Identity Corporation

5. Provide the license for PingIntelligence in `values.yaml`.

   The system prompts you for the following information from the license file:

```
#License required for ABS and ASE
 license : |
      ID=
      Organization=
      Product=
      Version=
      IssueDate=
      EnforcementType=
      ExpirationDate=
      MaxTransactionsPerMonth=
      Tier=
      SignCode=
      Signature=
```

6. Install PingIntelligence Helm-Chart.

   Helm-Chart creates Kubernetes secrets to store data of the released version in the namespace. The user can decide where to store them, in the default namespace in which PingIntelligence is being deployed.

   Choose from:

   - Create the namespace through Helm-Chart:

     1. Set `create_namespace` to `true` in `values.yaml`.

     2. Install by running the following command:

```
helm install -f values.yaml pi4api ~/pingidentity/helm-chart/pi4api
```

> **Note**
>
> This creates the namespace at deployment time, and the Helm-Chart release secret is stored in the default namespace.

   - Create the namespace before installation:

     1. Create a namespace:

```
kubectl create namespace pingidentity
```

     2. Install by running the following command:

```
helm install -f values.yaml pi4api ~/pingidentity/helm-chart/pi4api
```


Copyright © 2025 Ping Identity Corporation

> **Note**
>
> In this case, Helm-Chart creates a release secret that is stored in the namespace where PingIntelligence gets deployed.

> **Note**
>
> - Currently, PingIntelligence supports only image and license upgrades in Helm-Chart:
>
>   ```
>   helm upgrade -f values.yaml pi4api ~/pingidentity/helm-chart/
>   pi4api -n <namespace>
>   ```
>
> - Currently, PingIntelligence Helm-Chart supports a single node Kafka, Zookeeper, API Publish, and Mongo installation.

Example:

Copyright © 2025 Ping Identity Corporation

The following is an example of a default `values.yaml` file:

Copyright © 2025 Ping Identity Corporation

```
# Default values for pi4api kubernetes setup.
# This template is for example puprose
# Override these values in custom values.yaml
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

deployment :
 Namespace creation if required
 create_namespace: false
 #Namespace to deploy PI4API
 namespace : pingidentity
 run_user : 10001
 fsgroup_user : 101
 #Timzeone in which PI4API can be deployed.
 timezone : "utc"
 #Update Stratergy for pods.
 updateStrategy: RollingUpdate
 #License required for ABS and ASE
 license : |
     ID=
     Organization=
     Product=
     Version=
     IssueDate=
     EnforcementType=
     ExpirationDate=
     MaxTransactionsPerMonth=
     Tier=
     SignCode=
     Signature=
 #This is required to configure max_map_count in node running dashboard, if
set manually you can make enabled as false.
 dashboard_init:
   enabled: true
   repository: "busybox"
   tag: "latest"
   pullPolicy: "Always"
 #ABS configuration
 abs :
   image : pingidentity/abs:5.1
   terminationGracePeriodSeconds : 60
   replicas : 1
   #Port for ABS
   port : 8080
   health_api_path: /abs/health
   #ENVIRONMENT_VARIABLES
   environment_variables :
                # Access keys and secret keys to access ABS
                abs_access_key : "abs_ak"
                abs_secret_key : "abs_sk"
                abs_access_key_ru : "abs_ak_ru"
                abs_secret_key_ru : "abs_sk_ru"
                #Mongo url is passed here, you can configura external mongo
url .
```

Copyright © 2025 Ping Identity Corporation

```
                    mongo_rs: "mongodb://mongo-0.mongo-internal-service:27017"
                    # Communication between mongo and ABS
                    mongo_ssl : "true"
                    # Mongo DB Server Certificate Verification
                    # Set to true if Mongo DB instance is configured in SSL mode
and you want to do the server certificate verification
                    # By default ABS will not verify the MongoDB server
certificate
                    mongo_certificate : "false"
                    # Mongo DB User and password
                    mongo_username : "absuser"
                    mongo_password : "abs123"
                    # Duration of initial training period (units in hours)
                    # This value will be set in the mongo nodes
                    attack_initial_training : "24"
                    attack_update_interval : "24"
                    api_discovery_initial_period : "1"
                    api_discovery_update_interval : "1"
                    api_discovery : "true"
                    api_discovery_subpath : "1"
                    poc_mode: "true"
                    # Give the host:port combination of mutiple kafka server in
comma seperated.
                    kafka_server: kafka:9093
                    kafka_min_insync_replica: 1
                    #Users in Kafka for abs
                    abs_consumer_user: abs_consumer
                    abs_producer_user: abs_producer
                    abs_consumer_group: pi4api.abs
                    # Kafka Consumer Producer Password
                    abs_consumer_password: changeme
                    abs_producer_password: changeme
                    #topics to be created in kafka
                    transaction_topic: pi4api.queuing.transactions
                    attack_topic: pi4api.queuing.ioas
                    anomalies_topic: pi4api.queuing.anomalies
                    topic_partition: 1
                    replication_factor: 1
                    retention_period: "172800000"
                    attack_list_count: 100000
                    # Memory for webserver and streaming server (unit is in MB)
                    system_memory: 4096
                    # CLI admin password
                    abs_cli_admin_password: admin
                    # Configure Email Alert. Set enable_emails to true to
configure
                    # email settings for ABS
                    enable_emails: false
                    smtp_host: smtp.example.com
                    smtp_port: 587
                    smtp_ssl: true
                    smtp_cert_verification: false
                    sender_email: sender@example.com
                    sender_email_password: changeme
                    receiver_email: receiver@example.com
```


Copyright © 2025 Ping Identity Corporation

```
  pvc :
    volume_home_mount_path: /opt/pingidentity
    accessModes: ReadWriteOnce
    abs_data :
      pvc_type: gp2
      pvc_size: 100Gi
    abs_logs:
      pvc_type: gp2
      pvc_size: 10Gi
  resources:
      limits:
        cpu: "6"
        memory: 22G
      requests:
        cpu: "3"
        memory: 16G
  external_service :
    expose_external_service: false
    type : "LoadBalancer"
    port : 8080
 #API Publish deployment Configuration
 apipublish :
  image : pingidentity/apipublish:5.1
  replicas : 1
  #ports for the PingIntelligence API Publish Service
  port : 8050
  health_api_path: /apipublish/health
  environment_variables :
                # MongoDB Database names, Mongo url are picked from abs they
are same
                database_name: abs_data
                meta_database: abs_metadata
  resources:
      limits:
        cpu: "1"
        memory: 4G
      requests:
        cpu: "1"
        memory: 3G
  external_service :
    expose_external_service: false
    type : "LoadBalancer"
    port: 8050

#Mongo deployment Configuration
 mongo :
  # Flag to install mongo pod as part of setup, currently mongo cluster in
not supported , for using external mongo set it to false and put mongo url in
abs mongo_rs.
  install_mongo : true
  image : pingidentity/mongo:4.2.0
  replicas: 1
  port: 27017
  environment_variables :
```

Copyright © 2025 Ping Identity Corporation

```
                mongo_username : "absuser"
                mongo_password : "abs123"
                wired_tiger_cache_size_gb : "3"
                mongo_ssl : "true"
   pvc :
     volume_home_mount_path: /opt/pingidentity
     accessModes: ReadWriteOnce
     mongo_data :
       pvc_type: gp2
       pvc_size: 50Gi
     mongo_logs :
       pvc_type: gp2
       pvc_size: 50Gi
   resources:
           limits:
             cpu: "4"
             memory: 6G
           requests:
             cpu: "1"
             memory: 4G
   service :
     type: "ClusterIP"
   external_service :
     expose_external_service: false
     type : "LoadBalancer"
     port: 27017

 #Dashboard Deployment Configuration
 dashboard :
   image : pingidentity/dashboard:5.1
   replicas: 1
   webgui_port: 8030
   dataengine_port : 8040
   terminationGracePeriodSeconds : 60
   dataengine_health_api_path: /status
   environment_variables :
                #ENVIRONMENT_VARIABLES
                enable_syslog : "false"
                syslog_host : "127.0.0.1"
                syslog_port : "514"
                enable_attack_log_to_stdout : "true"
                ase_access_key : "admin"

                webgui_admin_password : "changeme"

                # User with permission set similar to "elastic" user, set
user if using external elastic search
                elastic_username: "elastic"

                # Passwords for "elasticsearch","ping_user" and "ping_admin"
users
                # dataengine will be accessible for these accounts
                # Please set strong passwords
                # If enable_xpack is set to false, below passwords are
ignored
```


Copyright © 2025 Ping Identity Corporation

```
                    elastic_password: "changeme"

                    # Configuration distribution type of elasticsearch. Allowed
values are default or aws
                    distro_type : "default"
                    # external elastic search url if not using internal elastic
search
                    elasticsearch_url: ""

                    #Users and passord in Kafka for dataengine
                    de_consumer_password : "changeme"
                    de_consumer_user: "pi4api_de_user"
                    de_consumer_group: "pi4api.data-engine"
                    # allowed values: native, sso.
                    # In native mode, webgui users are self managed and stored in
webgui.
                    # In sso mode, webgui users are managed and stored in an
Identity provider.
                    authentication_mode: native
                    # Maximum duration of a session.
                    # Value should be in the form of <number><duration_suffix>
                    # Duration should be > 0.
                    # Allowed duration_suffix values: m for minutes, h for hours,
d for days.
                    session_max_age: 6h

                    # Number of active UI sessions at any time.
                    # Value should be greater than 1.
                    max_active_sessions: 50

                    # webgui "ping_user" account password
                    webgui_ping_user_password: "changeme"

                     Below sso configuration properties are applicable in sso
authentication_mode only.
                    # Client ID value in Identity provider.
                    sso_oidc_client_id: pingintelligence
                    # Client Secret of the above Client ID.
                    sso_oidc_client_secret: changeme
                    # OIDC Client authentication mode.
                    # Valid values: BASIC, POST, or NONE
                    sso_oidc_client_authentication_method: BASIC
                    # OIDC Provider uri
                    # WebGUI queries <issuer-uri>/.well-known/openid-
configuration to get OIDC provider metadata
                    # issuer ssl certificate is not trusted by default. So import
issuer ssl certificate into config/webgui.jks
                    # issuer should be reachable from both back-end and front-end
                    sso_oidc_provider_issuer_uri: https://127.0.0.1:9031

                    # Place the sso provider issuer-certificate in the following
path => <installation_path>/pingidentity/certs/webgui/
                    # Name of the file should be => webgui-sso-oidc-provider.crt

                    # claim name for unique id of the user in UserInfo response
```

Copyright © 2025 Ping Identity Corporation

```
                # a new user is provisioned using this unique id value
                sso_oidc_provider_user_uniqueid_claim_name: sub
                # claim name for first name of the user in UserInfo response
                # either first name or last name can be empty, but both
should not be empty
                sso_oidc_provider_user_first_name_claim_name: given_name
                # claim name for last name of the user in UserInfo response
                # either first name or last name can be empty, but both
should not be empty
                sso_oidc_provider_user_last_name_claim_name: family_name
                # claim name for role of the user in UserInfo response
                sso_oidc_provider_user_role_claim_name: role
                # additional scopes in authorization request
                # multiple scopes should be comma (,) separated
                # openid,profile scopes are always requested
                sso_oidc_client_additional_scopes:
                 End of sso configuration

                # ssl key store password of webgui hosts
                server_ssl_key_store_password: "changeme"
                server_ssl_key_alias: webgui

                # local h2 db datasource properties
                h2_db_password: changeme
                h2_db_encryption_password: changeme

                # allowed values: abs/pingaccess/axway
                discovery_source: abs
                # allowed values: auto/manual
                discovery_mode: auto
                # value is in minutes
                discovery_mode_auto_polling_interval: 10
                discovery_mode_auto_delete_non_discovered_apis: false

                # valid only if discovery_source is set to pingaccess
                pingaccess_url: https://127.0.0.1:9000/
                pingaccess_username: Administrator
                pingaccess_password:

                # valid only if discovery_source is set to axway
                axway_url: https://127.0.0.1:8075/
                axway_username: apiadmin
                axway_password:
   pvc :
     volume_home_mount_path: /opt/pingidentity
     accessModes: ReadWriteOnce
     elasticsearch_data :
       pvc_type: gp2
       pvc_size: 50Gi
     webgui_data :
       pvc_type: gp2
       pvc_size: 50Gi
   resources:
           limits:
             cpu: "6"
```


Copyright © 2025 Ping Identity Corporation

```
            memory: 16G
          requests:
            cpu: "2"
            memory: 6G
   external_service :
     type : "LoadBalancer"
     https_Port : 443
 #ASE Deployment Configuration
 ase :
   image : pingidentity/ase:5.1
   # Define ports for the PingIntelligence API Security Enforcer
   http_ws_port: 8000
   https_wss_port: 8443
   management_port: 8010
   cluster_manager_port: 8020
   replicas : 1
   terminationGracePeriodSeconds : 60
   environment_variables :
                # Deployment mode for ASE. Valid values are inline or
sideband
                mode : "inline"
                enable_cluster : "false"
                enable_abs : "true"
                # Password for ASE keystore
                keystore_password: asekeystore
                # enable keepalive for ASE in sideband mode
                enable_sideband_keepalive : "false"
                enable_ase_health : "false"
                # Set this value to true, to allow API Security Enforcer to
fetch attack list from ABS.
                enable_abs_attack : "true"
                # Set this value to true, to allow API Security Enforcer to
fetch published API list from ABS
                enable_abs_publish : "false"
                #This value determines how often API Security Enforcer will
get published API list from ABS.
                abs_publish_request_minutes : 10
                # enable strict parsing checks for client requests
                # If enabled, ASE will block request with invalid header
start
                # If disabled, it will allow requests
                enable_strict_request_parser : "true"
                # cluster_secret_key for ASE cluster
                cluster_secret_key: yourclusterkey
                # CLI admin password
                ase_secret_key: admin
   pvc :
     volume_home_mount_path: /opt/pingidentity
     accessModes: ReadWriteOnce
     ase_data :
       pvc_type: gp2
       pvc_size: 10Gi
     ase_logs:
       pvc_type: gp2
       pvc_size: 100Gi
```

Copyright © 2025 Ping Identity Corporation

```
    ase_config:
      pvc_type: gp2
      pvc_size: 1Gi
  resources:
         limits:
           cpu: "4"
           memory: 8G
         requests:
           cpu: "1"
           memory: 4G
  external_service :
    expose_external_service: true
    type : "LoadBalancer"
    http_Port : 8000
    https_Port : 8443

 #Kafka Deployment Configuration
 kafka :
   # Flag to install kafka and zookeeper pod as part of setup,for using
external kafka set it to false and put kafka url in abs kafka_server.
   install_kafka : true
   image : pingidentity/kafka:5.1
   replicas : 1
   # Define ports for the Kafka
   sasl_port : 9093
   ssl_port: 9092
   environment_variables :
               #Zookeeper Service host:port.
               zookeeper_url: zookeeper:2182
               #Enable delete topic
               delete_topic: true
   resources:
         limits:
           cpu: "2"
           memory: 8G
         requests:
           cpu: "1"
           memory: 4G
   pvc :
     volume_home_mount_path: /opt/pingidentity
     accessModes: ReadWriteOnce
     data_volume :
       pvc_type: gp2
       pvc_size: 10Gi
     log_volume:
       pvc_type: gp2
       pvc_size: 1Gi
   external_service :
     expose_external_service: false
     type : "LoadBalancer"
     sasl_port : 9093
     ssl_port: 9092
  #Zookeeper Deployment Configuration
 zookeeper :
   image : pingidentity/zookeeper:5.1
```


Copyright © 2025 Ping Identity Corporation

```
replicas: 1
# Define ports for the Zookeeper
port : 2181
ssl_port: 2182
resources:
        limits:
          cpu: "2"
          memory: 8G
        requests:
          cpu: "1"
          memory: 4G
pvc :
  volume_home_mount_path: /opt/pingidentity
  accessModes: ReadWriteOnce
  data_volume :
    pvc_type: gp2
    pvc_size: 10Gi
  data_log_volume :
    pvc_type: gp2
    pvc_size: 100Gi
  log_volume:
    pvc_type: gp2
    pvc_size: 1Gi
external_service :
  expose_external_service: false
  type : "LoadBalancer"
  ssl_port: 2182
```

Next steps

Verify that the deployment is successful by entering the following command:

```
kubectl get pods -n pingidentity
```

Below is an example of what you should see:

```
NAME           READY   STATUS    RESTARTS   AGE
abs-0          1/1     Running   0          3d
apipublish-0   1/1     Running   1          3d
ase-0          1/1     Running   0          3d
dashboard-0    1/1     Running   0          3d
kafka-0        1/1     Running   0          3d
mongo-0        1/1     Running   0          3d
zookeeper-0    1/1     Running   0          3d
```

Fetch the IP addresses of ASE, ABS, and Dashboard by entering the following command:

```
kubectl get svc -n pingidentity
```

Copyright © 2025 Ping Identity Corporation

Below is an example of what you should see:

```
NAME                          TYPE           CLUSTER-IP       EXTERNAL-
IP
PORT(S)                         AGE
abs-internal-service          ClusterIP      None
<none>                                                                    8080/
TCP                        3d
apipublish-internal-service   ClusterIP      None
<none>                                                                    8050/
TCP                        3d
ase-external-service          LoadBalancer   10.100.249.102
a0f15298c7d7d42f183605d73258ebb1-2044570848.ap-northeast-2.elb.amazonaws.com   8000:30180/TCP,
8443:31961/TCP   3d
ase-internal-service          ClusterIP      None
<none>                                                                    8020/TCP,8010/
TCP               3d
dashboard-external-service    LoadBalancer   10.100.205.84
aa08fa369b08a4ed997a9371faf4418c-349939151.ap-northeast-2.elb.amazonaws.com    443:32068/
TCP                    3d
kafka                         ClusterIP      10.100.198.185
<none>                                                                    9092/TCP,9093/
TCP               3d
mongo-internal-service        ClusterIP      None
<none>                                                                    27017/
TCP                        3d
zookeeper                     ClusterIP      10.100.59.16
<none>                                                                    2182/TCP,2181/
TCP               3d
```

> **Note**
>
> If you are deploying in the sideband mode, take the NodePort IP address of ASE to use in API gateway integration.

PingIntelligence Hardening Guide

Accessing the PingIntelligence security hardening guide

The PingIntelligence for APIs security hardening guide provides administrators with a single point of reference for configurations and best practices available to harden their PingIntelligence for APIs platform.

Before you begin

You must have a registered account to the Ping Identity support and community portal.

About this task

Copyright © 2025 Ping Identity Corporation

One of the key security principles we follow at Ping is to make configurations secure by default. However, it is not always possible to create one-size-fits-all security configurations. This guide contains recommendations on how PingIntelligence administrators can further harden their platform based on their individual needs.

The recommendations are grouped by different PingIntelligence functional components. When deploying a component in PingIntelligence, see the corresponding section in the PingIntelligence security hardening guide.

To access to PingIntelligence security hardening guide:

Steps

- Go to the PingIntelligence security hardening guide and sign on to your Ping account.

Next steps

If you need further assistance, contact the Ping Identity Sales team.

PingIntelligence Monitoring Guide

This guide provides steps for monitoring the performance and health of your PingIntelligence deployment.

To ensure the optimal performance of your PingIntelligence deployment, you might need to conduct health checks on PingIntelligence components. You can also find recommendations on resource-utilization thresholds and patterns and troubleshooting tips.

PingIntelligence Health Check Guide

This section of the PingIntelligence Monitoring Guide provides administrators with a list of commands that can be used to perform health checks on different PingIntelligence components.

There are multiple methods explained for each component. You can automate the steps or use them in manual mode. The document also captures information on log files, process ID (PID) details, and port details of the API Security Enforcer (ASE) nodes, the API Behavioral Security (ABS) artificial intelligence (AI) engine, and the PingIntelligence Dashboard.

For more information, click the tab for the respective PingIntelligence components:

Copyright © 2025 Ping Identity Corporation

ASE

Performing health checks on ASE
About this task

You can use the following options to conduct a health check on ASE nodes:

Steps

- To enable the ASE health check URL in the `/pingidentity/ase/config/ase.conf` file, set the `enable_ase_health` config property to `true`.

  The default value of `enable_ase_health` is `false`.

  1. If the configuration is modified on a running ASE node, restart the node after modifying the configuration.

     For more information, see Starting and stopping ASE.

  2. In a clustered ASE environment, stop the ASE cluster and update the `ase.conf` file of the primary node and restart the other ASE nodes.

     For more information, see Restarting an ASE cluster.

  3. When the `enable_ase_health` is set to `true`, go to the following URLs and do a health check:

     - http://<*ase-hostname/ip*>:<*http_port*>/ase
     - https://<*ase-hostname/ip*>:<*https_port*>/ase

  If ASE is receiving the traffic, the response is `200 OK`.

- To check the status of an ASE process, the running status of HTTP or HTTPS process, and port number, run the `status` command-line interface (CLI) command.

  ```
  $./bin/cli.sh status
  ```

  This command also gives basic configuration information.

- To show the status of communication between ABS and all the ASE nodes in a cluster, run the following command:

  ```
  $ ./bin/cli.sh -u admin -p admin abs_info
  ```

  The `abs_info` command shows the last log upload and attack fetch information from ABS. If ASE is having any issues in uploading logs to ABS or connecting to ABS, it is reported in the output of the command.

- If ASE is running as a `systemctl` service, use the following command to check the status of the service:

  ```
  $ systemctl status pi-ase.service
  ```


Copyright © 2025 Ping Identity Corporation

ABS AI Engine

Performing health checks on the ABS AI engine

About this task

Use the following options to conduct a health check on the ABS AI engine:

Steps

- To check the ABS Admin API, use the ABS Admin REST API either from the Postman Collection or use the `curl` command:

```
$ curl -k -X GET 'https://<ABS Hostname>/IP:8080/v4/abs/admin' -H 'x-abs-ak:  <ABS access key>' -H 'x-abs-sk:  <ABS ssecret key>'
```

- If the ABS AI engine is running as a `systemctl` service, use the following command to check the status of the service:

```
$ systemctl status pi-abs.service
```

- To check the ABS log for job failures, use the following command:

```
$ grep allocated logs/abs/abs.log | grep failure
```

Troubleshooting:

If any failures are detected, reach out to the Ping Identity Support team.

- Check the ABS log for MongoDB heartbeats in the `/logs/abs/abs.log` file, which reports the status of MongoDB heartbeats at regular intervals.

  This file indicates any ABS to MongoDB connectivity issues.

Copyright © 2025 Ping Identity Corporation

PingIntelligence Dashboard

Performing health checks on the PingIntelligence Dashboard

About this task

Use the following commands to check the health status of the PingIntelligence Dashboard and its components:

Steps

- To check the health status of the Dashboard data engine:

  1. Run the `status` command to check the status of the Dashboard process:

     ```
     $ ./bin/cli.sh status
     ```

     It returns the status as `Running` or `Not Running`.

  1. If the Dashboard data engine is running as a `systemctl` service, use the following command to check the status of the service:

     ```
     $ systemctl status pi-data-engine
     ```

  2. To check the Dashboard log file for errors or exceptions, verify the `/pingidentity/dataengine/logs/admin/dataengine.log` file to detect connectivity issues between the Dashboard data engine and ABS or Elasticsearch:

     ```
     $ tail logs/admin/dataengine.log
     ```

- To check the health status of the WebGUI:

  1. To check if the WebGUI component is running, use the following health check URL in a browser or the curl command.

     Choose from:

     - The browser URL: https://<*WebGUI Hostname/IP*>:<*port*>/status
     - The curl command:

       ```
       $ curl -k -o /dev/null -s -w "%{http_code}\n" https://<webgui>:8030/status
       200
       ```

     A `200 OK` response results if the component is running.

  1. To show the status of the WebGUI process, run the `status` command:

     ```
     $ ./bin/cli.sh status
     ```


Copyright © 2025 Ping Identity Corporation

2. If the WebGUI is running as a `systemctl` service, use the following command to check the status of the service:

```
$ systemctl status pi-webgui.service
```

3. To check the WebGUI admin log file for errors or exceptions, verify the `/pingidentity/webgui/admin/logs/ admin.log` file to detect the connectivity issues between WebGUI and ABS or Elasticsearch:

```
$ tail logs/admin/admin.log
```

- To check the health status of Elasticsearch:

1. To check the health status of Elasticsearch using a health check URL:

   Choose from:

   - Using anonymous access:

     1. To enable access for anonymous user, add the following line to the `elasticsearch.yaml`:

        ```
        xpack.security.authc.anonymous.roles: monitoring_user
        ```

     You can update this during initial setup or later.

     1. If you are making the change on a running instance, restart Elasticsearch.
     2. After updating the `elasticsearch.yaml`, to check the status of Elasticsearch, go to https://<*Elasticsearch Hostname/IP*>:9200/.

        > **Note**
        >
        > You can use a browser or the following curl command:
        > +
        >
        > ```
        > $ curl -k -o /dev/null -s -w "%{http_code}\n" https://
        > <Elasticsearch Hostname/IP>:9200/
        > ```

     A `200 OK` response indicates a running Elasticsearch.

   - Using a health check user:

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> This approach does not require an Elasticsearch restart.

1. To add a health check user to Elasticsearch, run the following command:

```
curl -u elastic:<elastic user password>  -k -X POST "https://localhost:
9200/_xpack/security/user/<health_check_user>?pretty" -H 'Content-Type:
application/json' -d'
{
  "password" : "<password for health_check_user>",
  "roles": ["monitoring_user"]
}
'
```

2. After adding the health check user, to check the status of Elasticsearch, go to https://*<health_check_user>*:*<password>*@*<Elastcisearch hostname/IP>*:9200/.

> **Note**
>
> You can use a browser or the following curl command:
> +
> ```
> $ curl -k -o /dev/null -s -w "%{http_code}\n" https://
> <health_check_user>:<password>@<Elastcisearch hostname/IP>:9200/
> ```

A `200 OK` response indicates a running Elasticsearch.

- Using Elasticsearch username and password:
  - To query the health status of Elasticsearch using the elastic user and its password to see a more comprehensive output, which also reports the state of the cluster, run the following curl command:

```
$ curl -XGET -k -H 'content-type: application/json; charset=UTF-8' -u "elastic:<password>"
'https://<elasticsearch hostname/IP>:9200/_cluster/health?pretty'
```

1. To check the health status of Elasticsearch when it is running as a `systemctl` service, run the following command:

```
$ systemctl status pi-elasticsearch.service
```

2. To check the Elasticsearch log for errors or exceptions, verify the Elasticsearch log for any exceptions or errors by running the following command:

```
$ tail logs/elasticsearch.log
```


Copyright © 2025 Ping Identity Corporation

- To check the health status of Kibana:

1. To check the health status of Kibana using a health check URL:

   Choose from:

   - Using anonymous access:

     1. To enable access, add the following line to the `kibana.yaml`:

        ```
        status.allowAnonymous: true
        ```

     You can update this during initial setup or later.

     1. If you are making the change on a running instance, restart Kibana.
     2. After updating the `kibana.yaml`, to check the status, go to https://*<Kibana Hostname/IP>*:5601/pi/ui/dataengine/api/status.

        > **Note**
        >
        > You can use a browser or the following curl command:
        > +
        >
        > ```
        > $ curl -k -o /dev/null -s -w "%{http_code}\n" https://<Kibana
        > Hostname/IP>:5601/pi/ui/dataengine/api/status
        > ```

   A `200 OK` response indicates a running Kibana instance.

   - Using health check user:

     1. To add a health check user to Kibana, run the following command:

        ```
        curl -u elastic:<elastic user password>  -k -X POST "https://localhost:
        9200/_xpack/security/user/<health_check_user>?pretty" -H 'Content-Type:
        application/json' -d'
        {
          "password" : "<password for health_check_user>",
          "roles": ["monitoring_user"]
        }
        '
        ```

     2. After adding the health check user, to check the status of Kibana, go to https://*<health_check_user>*:*<password>*@*<Kibana hostname/IP>*:5601/pi/ui/dataengine/api/status.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> You can use a browser or the following curl command:
> +
> ```
> $ curl -k -o /dev/null -s -w "%{http_code}\n"https://
> <health_check_user>:<password>@<Kibana hostname/IP>:5601/pi/ui/
> dataengine/api/status
> ```

A `200 OK` response indicates a running Kibana.

1. To check the health status of Kibana when it is running as a `systemctl` service, run the following command to check the status of the service:

   ```
   $ systemctl status pi-kibana.service
   ```

2. To check Kibana log for errors or exceptions, verify the Kibana log for any exceptions or errors by running the following command:

   ```
   $ tail logs/kibana.log
   ```

Logs, port numbers, and process IDs

Review this supplementary information on log file details, important port numbers, and process ID (PID) information of PingIntelligence for APIs components.

Log files

The following table shows the main log files of PingIntelligence components.

Copyright © 2025 Ping Identity Corporation

ASE	ABS AI Engine	PingIntelligence PingIntelligenceDashboard
ASE access, management, and audit logs	ABS logs **Note** `abs.log` must be the first place for debugging any issues on the ABS. The log has information about each machine learning job on the host. All incoming communication from ASE or the PingIntelligence Dashboard or REST API requests are logged in this file. It also has a periodic log on heartbeat to MongoDB.	• **Dashboard data engine:** `/pingidentity/dataengine/logs/dataengine.log` • **WebGUI:** `/pingidentity/webgui/logs/admin.log` and `/pingidentity/webgui/logs/sso.log` • **Elasticsearch:** `/pingidentity/elasticsearch/logs/elasticsearch.log` • **Kibana:** `/pingidentity/kibana/logs/kibana.log`

Port numbers

The following table shows important port numbers used by PingIntelligence components.

ASE	ABS AI Engine	PingIntelligence PingIntelligenceDashboard
ASE ports	ABS ports	• The PingIntelligence Dashboard server: 8030. Port number 8030 should be exposed to public internet. Make sure that your organization's firewall allows access to this port. • Elasticsearch: 9200 • Kibana: 5601 • H2 database: 9092. H2 database is installed and runs as a part of the PingIntelligence Dashboard.

PID information

All PingIntelligence components have their respective PID files. Refer to these files for monitoring or for getting the PID information of the processes.

ASE	ABS AI Engine	PingIntelligence PingIntelligenceDashboard
The ASE PID file contains the PID for the controller process and the HTTP balancer and HTTPS balancer processes: `/pingidentity/ase/logs/ase.pid`	The `/pingidentity/abs/data/abs.pid` file contains the PID for the main ABS process.	There are separate PID files for the different components of the PingIntelligence Dashboard: • `/pingidentity/dataengine/data/dataengine.pid` • `/pingidentity/webgui/logs/webgui.pid` • `/pingidentity/elasticsearch/logs/elasticsearch.pid` • `/pingidentity/kibana/logs/kibana.pid`

Copyright © 2025 Ping Identity Corporation

PingIntelligence Upgrade Guide

This guide provides steps for upgrading your PingIntelligence deployment.

To ensure the optimal performance of your PingIntelligence deployment, you might need to conduct health checks on PingIntelligence components. You can also find recommendations on resource-utilization thresholds and patterns and troubleshooting tips.

PingIntelligence 5.2 introduces a number of new features and improvements. See the PingIntelligence 5.2 (September 2023) release notes for the summary of new features in PingIntelligence 5.2.

Upgrading MongoDB

To upgrade PingIntelligence, you must upgrade MongoDB from version 4.2 to 4.4 before proceeding to upgrade MongoDB to version 5.0.

Before you begin

1. Upgrading MongoDB from version 4.2 to 5.0 is a multi-step process. Before performing any upgrades, it's crucial to back up your MongoDB data.
2. Test that the upgraded system is working correctly with MongoDB 4.4 before upgrading to MongoDB 5.0.
3. Stop all API Behavioral Security (ABS), API Publish, and Dashboard components.
4. Ensure all replica set members are running MongoDB version 4.2.
5. Ensure the 4.2 replica set has `featureCompatibilityVersion` set to `4.2`:
    1. Connect to each replica set member and check `featureCompatibilityVersion` using the following command:

    ```
    db.adminCommand( {
    getParameter: 1,
    featureCompatibilityVersion: 1
    } )
    ```

    2. To set or update `featureCompatibilityVersion`, run the following command on the primary:

    ```
    db.adminCommand( { setFeatureCompatibilityVersion: "4.2" } )
    ```

6. Ensure that no replica set member is in `.mongodb.com/docs/v5.3/reference/replica-states///[ROLLBACK]` or `.mongodb.com/docs/v5.3/reference/replica-states///[RECOVERING]` state by running the `rs.status() command` command in the primary.
7. Prior to upgrading a member of the replica set, confirm that the member was cleanly shut down.
8. Make sure that the `abs_rs.js` file has actual MongoDB IPs (instead of `localhost/127.0.0.1`) for the replica set members.

About this task

Copyright © 2025 Ping Identity Corporation

To upgrade MongoDB version 4.2 to 5.0:

Steps

1. Upgrade secondary members of the replica set:
    1. Shutdown the secondary `mongo` instance.
    2. https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.4.15.tgz`mongo` build] and untar it.
    3. Replace the 4.2 binary with the 4.4 binary.
    4. Restart the member.
2. To step down the primary and force an election of a new primary, connect a `.mongodb.com/docs/v5.3/reference/program/mongod///[mongod]` shell to the primary and use `.mongodb.com/docs/v5.3/reference/method/rs.stepDown///[rs.stepDown()]`.
3. When `.mongodb.com/docs/v5.3/reference/method/rs.status///[rs.status()]` shows that the primary has stepped down and another member has assumed `PRIMARY` state, upgrade the stepped-down primary:
    1. Shut down the stepped-down primary and replace the `.mongodb.com/docs/v5.3/reference/program/mongod///[mongod]` binary with the 4.4 binary.
    2. Restart the member.
4. To enable backwards-incompatible 4.4 features, run the `setFeatureCompatibilityVersion` command on the primary in the admin database:

    ```
    db.adminCommand( { setFeatureCompatibilityVersion: "4.4" } )
    ```

5. Repeat the prerequisites and steps 1-4 to upgrade from 4.4.15 to 5.0.18.

Next steps

Complete the steps in Migrating MongoDB from RHEL 7.9 to 8.

Migrating MongoDB from RHEL 7.9 to 8

After upgrading MongoDB, migrate MongoDB from RHEL 7.9 to RHEL 8.

Before you begin

Make sure to configure the primary and secondary as a replica set in RHEL 7.9 and that the version is 5.0.18.

About this task

To migrate MongoDB from RHEL 7.9 to RHEL 8:

Steps

1. Install the MongoDB replica set on RHEL 8 instances.
    1. Create RHEL 8 new instances.

Copyright © 2025 Ping Identity Corporation

2. To create the MongoDB directory structure, create the following directories on each MongoDB node: `mongo`, `data`, `logs`, and `key`

```
mkdir -p /opt/pingidentity/mongo/data /opt/pingidentity/mongo/logs \ /opt/pingidentity/mongo/key
```

3. Download MongoDB 5.0.18 on each node and extract it:

```
cd /opt/pingidentity/
```

```
/opt/pingidentity# wget \ https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.2.0.tgz \ -O mongodb.tgz && tar xzf mongodb.tgz -C /opt/pingidentity/mongo/ --strip-components=1
```

4. Update the shell path variable and reload the shell:

```
/opt/pingidentity# echo PATH=$PATH:/opt/pingidentity/mongo/bin >> ~/.bashrc
```

```
/opt/pingidentity# source ~/.bashrc
```

5. Copy the `mongo-keyfile` and `mongodb.pem` files from the primary `mongo` and copy the files to the new `mongo` RHEL 8 instance key folder: `/opt/pingidentity/mongo/key`.

6. Optional: Copy the `abs_init.js` and `abs_rs.js` files from RHEL 7.9 `mongo` to RHEL 8 primary `mongo`.

7. Change the key file permissions.

```
/opt/pingidentity/mongo# chmod 600 key/mongodb-keyfile
```

8. Start `mongo` with TLS mode enabled using the same mode as the existing RHEL 7.9 `mongo` members.

```
mongod --auth --dbpath ./data/ --logpath ./logs/mongo.log --port 27017 --replSet absrs01 --fork --keyFile ./key/mongodb-keyfile -bind_ip 0.0.0.0 --tlsMode requireTLS --tlsCertificateKeyFile ./key/mongodb.pem
```

2. Add the newly created RHEL 8 `mongo` instances to the existing RHEL 7.9 `mongo` replica set.

1. Add the newly created `mongo` instances as a new replica set member to the existing RHEL 7.9 `mongo` setup using the following command in the RHEL 7.9 primary `mongo` shell:

```
rs.add( { host: “new_mongo_IP_1:27017" } )
```

```
rs.add( { host: “new_mongo_IP_2:27017" } )
```


Copyright © 2025 Ping Identity Corporation

2. Check the replica set members status by running the status command in primary `mongo`:

```
rs.status()
```

Result:

The initial status of the newly added replica members is `STARTUP2`. Once the initial sync completes, the newly added member's state changes to `SECONDARY`.

3. Set one of the newly created members as primary by executing the shell commands from the existing RHEL 7.9 primary `mongo`.

    1. Copy the replica set configuration to a variable:

```
cfg = rs.conf()
```

    2. Change each member's priority value based on which member you want as a `PRIMARY` and give highest priority to the node you want to set as a primary node.

```
cfg.members[0].priority = 1
cfg.members[1].priority = 1
cfg.members[2].priority = 1
cfg.members[3].priority = 100
cfg.members[4].priority = 100
cfg.members[5].priority = 100
```

> **Note**
>
> In the commands above, `members[0]`, `members[1]`, and `members[2]` are RHEL 7.9 `mongo` instances, and `members[3]`, `members[4]`, and `members[5]` are newly added RHEL 8 `mongo` instances.

    3. Assign the new `config` to the replica set:

```
rs.reconfig(cfg)
```

    4. Check the current status using the `rs.status()` command. Make sure one of the newly added RHEL 8 `mongo` instance is `PRIMARY`, and the rest are all `SECONDARY`.

3. Remove the RHEL 7.9 `mongo` members from the replica set.

    1. Run the following command from the current RHEL 8 primary `mongo`:

```
rs.remove("RHEL79_mongo_IP_1:27017")
rs.remove("RHEL79_mongo_IP_2:27017")
```

    2. Check the current status using the `rs.status()` command.

Result:

Copyright © 2025 Ping Identity Corporation

It should show only the RHEL 8 `mongo` members (one among them should be `PRIMARY`).

1. Change ABS and API Publish configurations to point to the new replica set (RHEL 8).
2. Shut down the RHEL 7.9 `mongo` instances.

Next steps

Complete the steps in Migrating Elasticsearch from RHEL 7.9 to 8.

Migrating Elasticsearch from RHEL 7.9 to 8

After migrating MongoDB, migrate Elasticsearch from RHEL 7.9 to RHEL 8.

Before you begin

Note the documents count before migration to compare the data after migration is complete:

```
curl -X GET "https://elasticsearch_ip:9200/_cat/indices?v" -u "elastic:changeme" -k
```

About this task

> **Note**
>
> Elasticsearch installation is allowed for non-root user (`ec2-user`).

To migrate Elasticsearch:

Steps

1. Create a new RHEL 8 instance with 8 core CPU, 16 GB, 1 TB hard disk drive (HDD).
2. Download and install JAVA 11.0.2 and set `JAVA_HOME` with the following command:

   ```
   export JAVA_HOME=/home/ec2-user/pingidentity/java
   export PATH=$JAVA_HOME/bin:$PATH
   ```

3. Download the Elasticsearch build version that you're migrating from.
4. Untar the Elasticsearch build by maintaining the same directory structure as the existing Elasticsearch:

   ```
   /opt/pingidentity/elasticsearch
   ```

5. Copy `elasticsearch.yml`, `elasticsearch_key.pem`, `elasticsearch_cert.pem`, and `elasticsearch.keystore` from RHEL 7.9 Elasticsearch to the corresponding path in the RHEL 8 Elasticsearch instance.

   > **Note**
   >
   > If the path is different, you'll need to update the path in the `.yml` file.

6. Start Elasticsearch.

Copyright © 2025 Ping Identity Corporation

```
./bin/elasticsearch -d -p ./logs/elasticsearch.pid
```

7. To verify that Elasticsearch started successfully, monitor the logs.
8. Stop both the RHEL 7 and RHEL 8 Elasticsearch process.
9. Copy the complete `data/` folder from the RHEL 7 instance to the new RHEL 8 instance.
10. Start Elasticsearch in the RHEL 8 instance.
11. Verify the documents count.
12. Point the `webgui` and `dataengine` to the new Elasticsearch IP by changing the Elasticsearch IP in `webgui.properties` and `dataengine.properties`.
13. Start the PingIntelligence 5.1 components that point to RHEL 8 MongoDB and Elasticsearch:
    1. Make sure that the API Behavioral Security (ABS) and API Publish configurations are pointing to the new `mongo` replica set (RHEL 8).
    2. Make sure that `webgui` and `dataengine` are pointing to the new RHEL 8 Elasticsearch.
    3. Start the PingIntelligence components in the following order:
        1. API Security Enforcer (ASE)
        2. ABS
        3. API Publish
        4. Data engine
        5. Web GUI
    4. To verify the data, access the Dashboard and verify the transactions count, indicators of attack (IOA), and discovered APIs.
    5. Make an `h2-backup`.
        1. Create an `h2backup.sh` file inside the `webgui/bin` folder with the following data:

Copyright © 2025 Ping Identity Corporation

```
#!/bin/bash
DATABASE_USER="sa"
DATABASE_PASSWORD="changeme changeme"
DATABASE_URL="jdbc:h2:ssl://localhost/webgui_data;CIPHER=AES"

H2_JAR_PATH="/home/ec2-user/pingidentity/webgui/lib/external/h2-*.jar"
TRUST_STORE_PATH="/home/ec2-user/pingidentity/webgui/config/webgui.jks"
OUTPUT_DIRECTORY="/home/ec2-user/pingidentity/webgui/data/h2-backup"

# Function to display an error message and exit with a non-zero status
function exit_with_error() {
    echo "Error: $1"
    exit 1
}

# Check if required environment variables are set
if [ -z "$DATABASE_USER" ] || [ -z "$DATABASE_PASSWORD" ] || [ -z "$DATABASE_URL" ] ||
[ -z "$H2_JAR_PATH" ] || [ -z "$TRUST_STORE_PATH" ] || [ -z "$OUTPUT_DIRECTORY" ]; then
    exit_with_error "One or more required environment variables are not set."
fi

# Ensure the H2 JAR file exists
if ! ls $H2_JAR_PATH > /dev/null 2>&1; then
    exit_with_error "H2 JAR file not found at '$H2_JAR_PATH'."
fi

# Ensure the trust store file exists
if [ ! -f "$TRUST_STORE_PATH" ]; then
    exit_with_error "Trust store file not found at '$TRUST_STORE_PATH'."
fi

# Ensure the output directory exists
if [ ! -d "$OUTPUT_DIRECTORY" ]; then
    exit_with_error "Output directory '$OUTPUT_DIRECTORY' not found."
fi

# List of tables to export
TABLES=("API_GROUP" "API_GROUP_ASSOCIATION" "API_STATE" "ASE_API" "CLIENT_VISIBILITY"
"DISCOVERY_API" "DISCOVERY_METADATA" "USER" "USER_SESSION")

for TABLE_NAME in ${TABLES[@]}; do
    # Generate the export query for the current table
    EXPORT_QUERY="SELECT * FROM $TABLE_NAME;"

    csv_path="$OUTPUT_DIRECTORY/$TABLE_NAME.csv"

    # Export the data of the current table as CSV
    java -Djavax.net.ssl.trustStore=$TRUST_STORE_PATH -cp $H2_JAR_PATH
org.h2.tools.Shell -url "$DATABASE_URL" -user "$DATABASE_USER" -password
"$DATABASE_PASSWORD" -sql "CALL CSVWRITE('$csv_path', '$EXPORT_QUERY')"

    # Check the exit status of the java command
    if [ $? -ne 0 ]; then
```


Copyright © 2025 Ping Identity Corporation

```
        exit_with_error "Failed to export data for table '$TABLE_NAME'."
    fi
done

echo "Data export completed successfully."
```

2. Update the paths for the following variables:

```
H2_JAR_PATH
TRUST_STORE_PATH
OUTPUT_DIRECTORY
```

3. Create the `h2-backup` directory under `data/`.

```
mkdir webgui/data/h2-backup
```

4. Give executable permissions to the `h2backup.sh` file.
5. Run the `./bin/h2backup.sh` script to create the `h2-backup` folder inside the `webgui/data` directory.
6. After successful completion of the script run, verify the backup data by checking the `.csv` files created in the `h2-backup` folder.

```
ls webgui/data/h2-backup/
API_GROUP.csv  API_GROUP_ASSOCIATION.csv  API_STATE.csv  ASE_API.csv
CLIENT_VISIBILITY.csv  DISCOVERY_API.csv  DISCOVERY_METADATA.csv  USER.csv
USER_SESSION.csv
```

7. Change `"discovery_source","abs"` to `"discovery_source","ABS"` in the `DISCOVERY_METADATA.csv` file, and save the file.

```
"seedDataInitAdmin","2023-09-07 17:00:31.365","discovery_source","ABS"
```

8. Copy and keep the `h2-backup` folder separate.

   The backup folder will be used in Upgrading PingIntelligence.

   Troubleshooting:

   If the error below is observed in the `webgui admin.log`, restart the Dashboard to resolve the issue.

```
error c.p.p.c.ElasticSearchClientConfiguration [https-jsse-nio-0.0.0.0-8030-exec-3]
FailureListener | Error with Node: [host=https://10.96.6.196:9200]
2023-08-22 06:22:00 error c.p.p.w.s.d.MainDashboardActionsServiceImpl [https-jsse-
nio-0.0.0.0-8030-exec-3] Failed to Fetch Attack Lists Count from Elasticsearch. Error:
pi4api.commons | Error while executing Elasticsearch msearch Query Type: bool.
com.pingidentity.pingintelligence.exception.PIOperationsException: pi4api.commons |
Error while executing Elasticsearch msearch Query Type: bool
```

Copyright © 2025 Ping Identity Corporation

Next steps

Complete the steps in Upgrading PingIntelligence.

Upgrading PingIntelligence

After upgrading Elasticsearch, upgrade PingIntelligence 5.1 to 5.2 and switch to RHEL 8.

Before you begin

Stop all PingIntelligence 5.1 components before starting the upgrade.

About this task

To upgrade PingIntelligence:

Steps

1. Upgrade API Security Enforcer (ASE) from 5.1.1 to 5.1.3 in the corresponding RHEL 7.9 instance.

   > **Note**
   > There is no 5.2 RHEL 8 build for ASE.

   1. Make sure that ASE is stopped.
   2. Make a backup of the existing ASE base folder.
   3. Copy the ASE 5.1.3 build.
   4. Untar the new build.
   5. Update the `ase.conf` and `abs.conf` with the required details, such as port, ASE mode, and API Behavioral Security (ABS) IP, by referring to backed-up `conf` files.
   6. Add the actual passwords for the following passwords:

      ```
      config/ase.conf: sender_password, keystore_password
      config/abs.conf: access_key, secret_key
      config/cluster.conf: cluster_secret_key
      ```

   7. Copy `ase.crt` from the backup folder to the `ase/config` directory.
   8. Copy the `PingIntelligence.lic` license file to the `ase/config` directory.
   9. Generate the master key.

      ```
      /opt/pingidentity/ase/bin/cli.sh generate_obfkey -u admin -p
      ```

   10. Obfuscate the key.

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh obfuscate_keys -u admin -p
```

11. Copy the API JSON files to the `config/api` directory.
12. Start ASE.

```
/opt/pingidentity/ase/bin/start.sh
```

2. In the already migrated RHEL 8 `mongo`, add the new DB `pi4api_dashboard` and grant `readWrite` roles for the `absuser` for this DB.

> **Note**
>
> Make sure the `pi4api_dashboard` collection does not exist before the 5.2 upgrade. The 5.2 upgrade will remove the data from the following tables, which are under the `pi4api_dashboard` collection, if it exists:
>
> ```
> api_groups
> api_state
> user_sessions
> users
> ```

1. Shut down the `mongo` primary and secondary.

```
mongod --shutdown --dbpath data/
```

2. Start the primary `mongo` without the `--auth` flag.

```
mongod --dbpath ./data/ --logpath ./logs/mongo.log --port 27017 --replSet absrs01 --fork -
bind_ip 0.0.0.0
```

3. Sign on to `mongo` without specifying a user.

```
mongo
```

4. Run the following commands:

```
create pi4api_dashboard DB
 use pi4api_dashboard
```

```
Switch to admin db
 use admin
```

Execute the command to grant the `readWrite` role for `absuser` for `pi4api_dashboard`.

Copyright © 2025 Ping Identity Corporation

```
db.grantRolesToUser("absuser", ["readWrite",{ role: "readWrite", db: "pi4api_dashboard" }]);
```

5. Shut down the `mongo` primary.
6. Restart `mongo` (both primary and secondary) with `--auth` enabled and with `--tlsMode`.

```
mongod --auth --dbpath ./data/ --logpath ./logs/mongo.log --port 27017 --replSet absrs01 --
fork --keyFile ./key/mongodb-keyfile -bind_ip 0.0.0.0 --tlsMode requireTLS --
tlsCertificateKeyFile ./key/mongodb.pem
```

3. Make the following changes in Kafka:
   1. Create the discovery topic.

```
/home/ec2-user/pingidentity/kafka/bin/kafka-topics.sh --bootstrap-server 172.16.40.244:9091 --
create --topic pi4api.queuing.apis --partitions 1 --replication-factor 1 --command-config /
home/ec2-user/pingidentity/kafka/config/client.properties
```

   2. Create the access control lists (ACL) for the ABS producer user for the discovery topic.

```
/home/ec2-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091 --
add --allow-principal User:abs_producer --operation Create --operation Read --operation Write
--topic pi4api.queuing.apis --command-config /home/ec2-user/pingidentity/kafka/config/
client.properties
```

   3. Create the ACLs for the ABS consumer user for the discovery topic.

```
/home/ec2-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091 --
add --allow-principal User:abs_consumer --operation Read --topic pi4api.queuing.apis --
command-config /home/ec2-user/pingidentity/kafka/config/client.properties
```

   4. Create the ACLs for the data engine consumer user.

```
/home/ec2-user/pingidentity/kafka/bin/kafka-acls.sh --bootstrap-server 172.16.40.244:9091 --
add --allow-principal User:pi4api_de_user --operation Create --operation Read --operation
Write --topic pi4api.queuing.apis --command-config /home/ec2-user/pingidentity/kafka/config/
client.properties
```

   5. Add the ACLs below in Kafka if they have not already been added:

Copyright © 2025 Ping Identity Corporation

```
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.anomalies,
patternType=LITERAL):
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=GROUP, name=pi4api.abs, patternType=LITERAL):
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.ioas,
patternType=LITERAL):
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.apis,
patternType=LITERAL):
        (principal=User:abs_producer, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=WRITE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=CREATE, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=CREATE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=GROUP, name=pi4api.data-engine,
patternType=LITERAL):
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
Current ACLs for resource ResourcePattern(resourceType=TOPIC, name=pi4api.queuing.transactions,
patternType=LITERAL):
        (principal=Group:pi4api.abs, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=User:pi4api_de_user, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=READ, permissionType=ALLOW)
        (principal=User:abs_consumer, host=, operation=DESCRIBE, permissionType=ALLOW)
        (principal=Group:pi4api.abs, host=, operation=READ, permissionType=ALLOW)
```

Copyright © 2025 Ping Identity Corporation

```
(principal=Group:pi4api.data-engine, host=, operation=DESCRIBE, permissionType=ALLOW)
(principal=User:pi4api_de_user, host=, operation=DESCRIBE, permissionType=ALLOW)
(principal=User:abs_producer, host=, operation=DESCRIBE, permissionType=ALLOW)
(principal=Group:pi4api.data-engine, host=, operation=READ, permissionType=ALLOW)
(principal=User:abs_producer, host=, operation=WRITE, permissionType=ALLOW)
```

4. Upgrade the Dashboard from 5.1.0.2 to 5.1.1.

> **Note**
>
> Make sure the `discovered_apis` index does not exist before upgrading the Dashboard from 5.1.0.2. During the 5.2 Dashboard upgrade, all the documents related to this index will be cleaned up.

1. Stop the `dataengine` and `webgui`.
2. Make a backup of `dataengine.jks`, `kafka_truststore.jks`, and `webgui.jks` files, and save them separately.
3. Delete the `dataengine` and `webgui` folders inside the Ping Identity directory.
4. Download the Dashboard 5.1.1 build to the Dashboard instance `/home/ec2-user` folder.
5. Untar the downloaded build.

Result:

The `dataengine` and `webgui` folders will be created inside the Ping Identity directory.

1. Make the following changes to `dataengine`:
    1. Go to the `dataengine` folder.

    ```
    cd /home/ec2-user/pingidentity/dataengine/config
    ```

    2. Copy the `dataenine.jks` (from the backup) file to the `dataengine/config` directory.
    3. Copy `kafka_truststore.jks` (from the backup) file to the `dataengine/config` directory.
    4. Update the properties below in `kafka.properties` with valid entries:

    ```
    pi.kafka.bootstrap-servers=10.96.6.45:9093
    pi.kafka.consumer.sslTruststoreLocation=/home/ec2-user/pingidentity/dataengine/config/
    kafka_truststore.jks
    pi.kafka.consumer.sslTruststorePassword=changeme
    pi.kafka.consumer.groupId=pi4api.data-engine
    pi.kafka.consumer.authentication.username=pi4api_de_user
    pi.kafka.consumer.authentication.password=changeme
    ```

    5. Update `dataengine.properties` with valid entries:

Copyright © 2025 Ping Identity Corporation

```
pi.dataengine.server.ssl.key-store-password=changeme
pi.dataengine.server.ssl.key-alias=<alias-name>
pi.dataengine.abs.url=https://10.96.6.82:8080
pi.dataengine.abs.access_key=abs_ak
pi.dataengine.abs.secret_key=abs_sk
pi.dataengine.elasticsearch.url=https://10.96.6.45:9200
pi.dataengine.elasticsearch.username=elastic
pi.dataengine.elasticsearch.password=changeme
```

2. Generate a new master key.

```
cd /home/ec2-user/dataengine
./bin/cli.sh generate_obfkey
```

3. Obfuscate keys.

```
./bin/cli.sh obfuscate_keys
```

4. Start the `datenegine`.

```
./bin/start.sh
```

5. Make the following changes to `webgui`:

    1. Go to the `webgui` folder.

    ```
    cd /home/ec2-user/pingidentity/webgui/config
    ```

    2. Copy the `webgui.jks` (from the backup) file to the `webgui/config` directory.

    3. Update the properties below in the `webgui.properties` file:

Copyright © 2025 Ping Identity Corporation

```
pi.webgui.server.ssl.key-store-password=changeme
pi.webgui.server.ssl.key-alias=<alis-name>
pi.webgui.abs.url=https://10.96.6.82:8080
pi.webgui.abs.api-service-url=https://10.96.6.82:8050
pi.webgui.abs.access-key=abs_ak
pi.webgui.abs.secret-key=abs_sk
pi.webgui.ase.url=https://10.96.6.80:8010
pi.webgui.ase.access-key=ase_ak
pi.webgui.ase.secret-key=ase_sk
pi.webgui.elasticsearch.url=https://10.96.6.45:9200
pi.webgui.elasticsearch.username=elastic
pi.webgui.elasticsearch.password=changeme
pi.webgui.datasource.username=sa
pi.webgui.datasource.password=changeme
pi.webgui.datasource.encryption-password=changeme
```

4. Generate a new master key.

```
cd /home/ec2-user/webgui
./bin/cli.sh generate_obfkey
```

5. Obfuscate keys.

```
./bin/cli.sh obfuscate_keys
```

6. Start the `webgui`.

```
./bin/start.sh
```

> **Note**
>
> Use only RHEL 8 instances to run the remaining PingIntelligence 5.2 components (ABS, API Publish, machine learning (ML) service, data engine, and web GUI).

5. Stop all PingIntelligence 5.1 components (ABS, API Publish, Dashboard) before starting the 5.2 upgrade.
6. To upgrade ABS, make sure you have a RHEL 8 instance ready to install the PingIntelligence ABS 5.2 build and then proceed with the following:
    1. Install Java 11.0.2 and set `JAVA_HOME`.
    2. Download the PingIntelligence ABS 5.2 build from the PingIntelligence Downloads website.
    3. Copy the build to the RHEL 8 instance.
    4. Untar the build.

       Result:

       An `abs` folder will be created inside the `pingidentity` folder.

Copyright © 2025 Ping Identity Corporation

5. Copy the `PingIntelligence.lic` license file to the `pingidentity/abs/config` directory.
6. Copy the `abs.jks` file from the old ABS (RHEL 7, ABS 5.1) to the new RHEL 8 ABS `ssl` directory.

```
/pingidentity/abs/config/ssl/
```

You can alternatively create a new `abs.jks` file.

1. Copy the `kafka.truststore.jks` file from the old ABS (RHEL 7, ABS 5.1) to new RHEL 8 ABS corresponding path.

```
/opt/pingidentity/abs/config/kafka.truststore.jks
```

2. Update the properties below in `Kafka.properties`:

```
pi.kafka.bootstrap-servers=10.96.6.196:9093
pi.kafka.sslTruststoreLocation=/opt/pingidentity/abs/config/kafka.truststore.jks
pi.kafka.sslTruststorePassword=<actual_password>
pi.kafka.consumer.authentication.password=<actual_password>
pi.kafka.producer.authentication.password=<actual_password>
pi.kafka.producer.min-insync-replicas=1
```

3. Update the `abs.properties` file with the details below:

```
jks_password=<actual_password>
Mongo_rs=mongodb://10.96.6.242:27017,10.96.6.201:27017
mongo_username=absuser
mongo_password=abs123
mongo_ssl=true
email_password=<actual_password>
```

4. Generate a new ABS master key.

```
/opt/pingidentity/abs/bin/cli.sh generate_obfkey -u admin -p admin
```

5. Obfuscate keys.

```
/opt/pingidentity/abs/bin/cli.sh obfuscate_keys -u admin -p admin
```

Result:

The following keys will be obfuscated:

`config/abs.properties`: `mongo_password`, `jks_password`, and `email_password`

`config/kafka.properties`: `pi.kafka.consumer.authentication.password`, `pi.kafka.producer.authentication.password`, and `pi.kafka.sslTruststorePassword`

6. Start ABS.

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/abs/bin/start.sh
```

7. To upgrade API Publish, make sure you have a RHEL 8 instance ready to install the PingIntelligence ABS 5.2 build and then proceed with the following:

    1. Install Java 11.0.2 and set `JAVA_HOME`.
    2. Download the PingIntelligence API Publish 5.2 build from the PingIntelligence Downloads website.
    3. Copy the build to the RHEL 8 instance.
    4. Untar the build.

       Result:

       An `apipublish` folder will be created inside the `pingidentity` folder.
    5. Copy the `apipublish.jks` file from the old API Publish (RHEL 7, API Publish 5.1) to the new RHEL 8 API Publish `ssl` directory.

       ```
       /pingidentity/apipublish/config/ssl/
       ```

    You can alternatively create a new `apipublish.jks` file.

    1. Update the `apipublish.properties` file with the details below:

       ```
       pi.apipublish.ssl.key-store-password=api123
       pi.apipublish.datasource.mongo_rs=mongodb://10.96.6.242:27017,10.96.6.201:27017
       pi.apipublish.datasource.username=absuser
       pi.apipublish.datasource.password=abs123
       pi.apipublish.datasource.mongo_ssl=true
       ```
    2. Generate a new API Publish master key.

       ```
       /pingidentity/apipublish/bin/cli.sh generate_obfkey -u admin -p admin
       ```
    3. Obfuscate keys.

       ```
       /pingidentity/apipublish/bin/cli.sh obfuscate_keys -u admin -p admin
       ```

       Result:

       The following keys will be obfuscated:

       `config/apipublish.properties`: `pi.apipublish.ssl.key-store-password` and `pi.apipublish.datasource.password`
    4. Start API Publish.

Copyright © 2025 Ping Identity Corporation

```
../bin/start.sh
```

8. Install the new ML service 5.2 build on the RHEL 8 instance by following the steps in Installing the PingIntelligence machine learning service.

9. To install `dataengine`, make sure you have a RHEL 8 instance with 8 core CPU, 16 GB, 1 TB hard disk drive (HDD).

    1. Download the PingIntelligence 5.2 Dashboard build and extract it in the RHEL 8 instance.

    2. Install Java 11.0.2 and set `JAVA_HOME`.

    3. Copy the `data-engine.jks` file from the old `dataengine` and copy it to the new `dataengine` RHEL 8 instance in `dataengine/config directory`.

    4. Add the Mongo certificate to `data-engine.jks`.

        1. In the RHEL 8 `mongo` primary node, go to `mongo/key/mongo.pem` and copy the public key part.

        2. Store the public key as `mongo.crt` in `dataengine/config`.

        3. Run the following command:

        ```
        keytool -import -keystore dataengine.jks -storetype JKS -storepass changeme -alias mongo
        -file mongo.crt -noprompt
        ```

    5. Copy the `kafka.truststore.jks` file to the `dataengine/config/` directory.

    6. Update `Kafka.properties` with the details below:

    ```
    pi.kafka.bootstrap-servers=<Kafka_IP>:9093
    pi.kafka.consumer.sslTruststoreLocation=/opt/pingidentity/dataengine/config/
    kafka_truststore.jks
    pi.kafka.consumer.sslTruststorePassword=<actual_password>
    pi.kafka.consumer.authentication.password=<actual_password>
    ```

    7. Update `Dataengine.properties` with the details below:

    ```
    pi.dataengine.server.ssl.key-store-password=<actual_password>
    pi.dataengine.server.ssl.key-alias=<alias-name>
    # abs properties
    pi.dataengine.abs.url=https://<ABS_IP>:8080
    pi.dataengine.abs.access_key=abs_ak
    pi.dataengine.abs.secret_key=abs_sk
    pi.dataengine.elasticsearch.url=https://<elasticsearch_ip>:9200
    pi.dataengine.elasticsearch.username=elastic
    pi.dataengine.elasticsearch.password=<actual_password>

    pi.dataengine.datasource.url=mongodb://<mongo_ip>:27017
    pi.dataengine.datasource.username=absuser
    pi.dataengine.datasource.password=abs123
    ```

Copyright © 2025 Ping Identity Corporation

8. Generate `dataengine_master.key`.

```
./bin/cli.sh generate_obfkey
```

9. Obfuscate keys.

```
./bin/cli.sh obfuscate_keys
```

10. Start `dataengine`.

```
./bin/start.sh
```

10. Install `webgui`.

1. Copy the `h2-backup` folder (that was copied and saved in Migrating Elasticsearch from RHEL 7.9 to 8 in step 13e) to the RHEL 8 instance under the `webgui/data` directory.
2. Copy `webgui.jks` from the old `webgui` instance to the RHEL 8 instance `webgui/config` directory.
3. Add the Mongo certificate to `webgui.jks`.
    1. In the RHEL 8 `mongo` primary node, go to `mongo/key/mongo.pem` and copy the public key part.
    2. Store the public key as `mongo.crt` in `webgui/config`.
    3. Run the following command:

```
keytool -import -keystore webgui.jks -storetype JKS -storepass changeme -alias mongo -
file mongo.crt -noprompt
```

4. Update `webgui.properties` with the details below:

Copyright © 2025 Ping Identity Corporation

```
pi.webgui.server.ssl.key-store-password=<actual_password>
pi.webgui.server.ssl.key-alias=<alias-name>

pi.webgui.abs.url=https://10.96.6.242:8080
pi.webgui.abs.api-service-url=https://10.96.6.242:8050
pi.webgui.abs.access-key=<actual_key>
pi.webgui.abs.secret-key=<actual_key>

# ase properties
pi.webgui.ase.url=https://10.96.6.217:8010
pi.webgui.ase.access-key=<actual_key>
pi.webgui.ase.secret-key=<actual_key>

# elasticsearch properties
pi.webgui.elasticsearch.url=https://10.96.6.19:9200
pi.webgui.elasticsearch.username=elastic
pi.webgui.elasticsearch.password=<actual_password>

pi.webgui.datasource.url=mongodb://10.96.6.242:27017
pi.webgui.datasource.username=absuser
pi.webgui.datasource.password=abs123
```

5. Generate `dataengine_master.key`.

```
./bin/cli.sh generate_obfkey
```

6. Obfuscate keys.

```
./bin/cli.sh obfuscate_keys
```

7. Start `webgui`.

```
./bin/start.sh
```

Copyright © 2025 Ping Identity Corporation

PingIntelligence Integrations

You can integrate PingIntelligence with a variety of API platforms.

For information about integrating PingIntelligence with an API gateway, see:

- Akana API gateway sideband integration
- Apigee integration
- AWS API Gateway integration
- Axway sideband integration
- Azure APIM sideband integration
- CA API gateway sideband integration
- F5 BIG-IP integration
- IBM DataPower Gateway sideband integration
- Kong API gateway integration
- MuleSoft sideband integration
- NGINX sideband integration
- NGINX Plus sideband integration
- PingAccess sideband integration
- PingFederate sideband integration
- TIBCO integration
- WSO2 integration

Akana API gateway sideband integration

This integration guide discusses PingIntelligence for APIs deployment in a sideband configuration with the Akana API Gateway.

PingIntelligence for APIs in a sideband deployment mode integrates with the Akana API Gateway to provide in-depth analytics on API traffic. A PingIntelligence policy is installed in the Policy Manager component of the Akana API Gateway to pass API metadata to PingIntelligence for detailed API activity reporting and attack detection. For more information on sideband deployment, see Sideband ASE.

PingIntelligence for APIs provides the JavaScript policy that extracts API metadata from a request and response processed by the Akana API Gateway. The API metadata is passed to API Security Enforcer (ASE). The following are a few highlights of the integration solution:

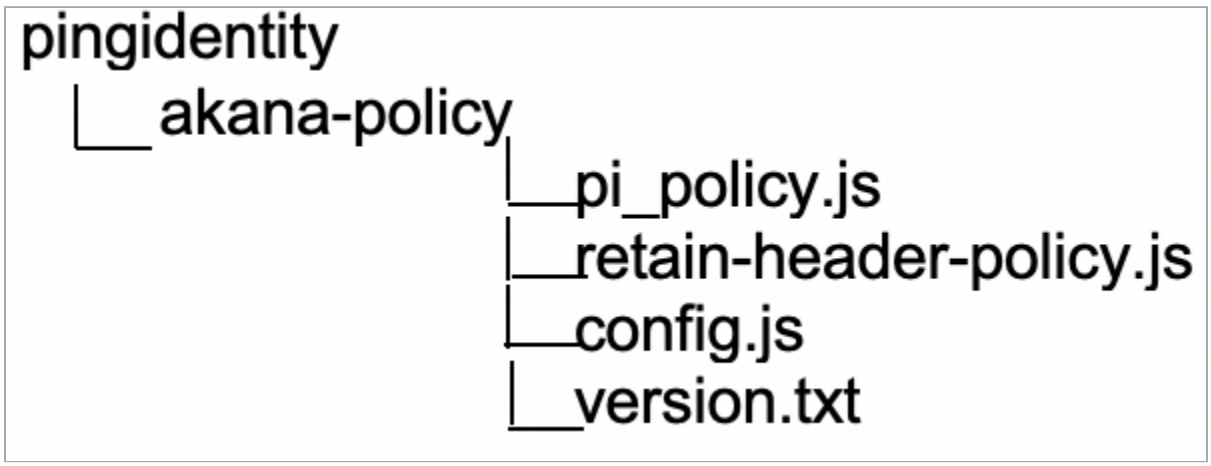
- Support for SSL connectivity through a valid certificate authority (CA)-signed certificate.
- Support for connection keep alive between the Akana Gateway and ASE for faster processing of request and response data.

Copyright © 2025 Ping Identity Corporation

- Support for ASE-failover by provisioning a secondary ASE.
- OAuth attribute extraction and username support for OAuth-enabled APIs.
- Interception of OAuth tokens sent as part of query parameters.

> **Note**
>
> The Akana Gateway does not support self-signed certificates.

Three PingIntelligence policies are made available to support the integration. The policies are packaged in the `pi-api-akana-policy-4.x.x.tar.gz` file. The following diagram shows the directory structure for reference.



`pi_policy.js`: This is the main PingIntelligence policy. It extracts the metadata for each API call, formats it into JSON and makes API calls to pass the metadata to ASE.

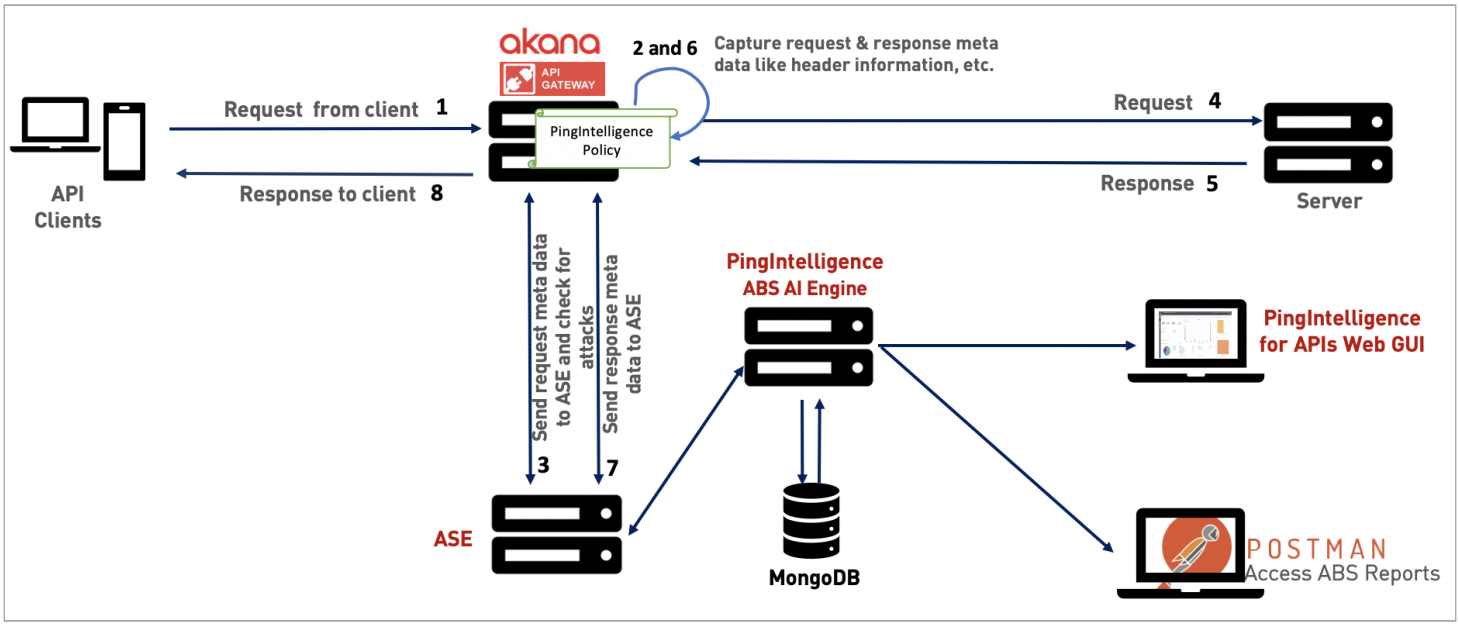
`retain-header-policy.js`: After validating a token with the OAuth server, Akana gateway deletes the incoming Authorization header. As a result, this header does not get forwarded to ASE. The `retainHeader.js` remedies this by capturing the deleted Authorization header and passes it to `pi_policy.js` for metadata extraction. The `retainHeader.js` policy gets executed before `pi_policy.js`.

`config.js`: This script takes ASE configuration as input from the user. The script then connects the ASE nodes and the policy.

> **Note**
>
> The `retain-header.js` policy needs to be attached to all OAuth-enabled APIs to ensure user information is extracted from API reqeusts.

The following diagram shows the logical setup of PingIntelligence for APIs components and the Akana API Gateway:

Copyright © 2025 Ping Identity Corporation



The traffic flow through the Akana API Gateway and PingIntelligence for APIs components is explained below:

1. The client sends an incoming request to Akana API gateway.
2. PingIntelligence policy deployed on the Akana API Gateway is executed on the request to extract the metadata from the incoming request.
3. Akana API gateway makes an API call to send the request metadata to API Security Enforcer (ASE). The ASE checks the client identifiers, such as usernames and tokens, against the deny list. If all checks pass, ASE returns a `200-OK` response to the Akana API gateway. If not, a different response code is sent to the Akana API Gateway (400 or 403). The request information is also logged by ASE and sent to the PingIntelligence API Behavioral Security (ABS) artificial intelligence (AI) engine for processing.
4. The Akana API gateway forwards the API requests to the backend server after the ASE processes it. If the gateway receives a `403-Forbidden` response from ASE, it blocks the client. Otherwise, it forwards the request to the backend server.
5. The response from the backend server is received by the Akana API Gateway.
6. The PingIntelligence policy is again applied on the response to extract the metadata from the server response.
7. The Akana API gateway makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing. ASE sends a `200-OK` to the API Gateway.
8. The Akana API gateway sends the response received from the backend server to the client.

Adding PingIntelligence ASE APIs

Add a primary and secondary ASE node to the Akana API Gateway.

Before you begin

Copyright © 2025 Ping Identity Corporation

You must:

- Install and configure the PingIntelligence software. For more information, refer to Automated deployment or Manual deployment.
- Verify that API Security Enforcer (ASE) is in sideband mode by running the following ASE command:

```
/opt/pingidentity/ase/bin/cli.sh status
API Security Enforcer
status                  : started
mode                    : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : enabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

  - If ASE is not in sideband mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set mode as sideband and start ASE.

- For a secure communication between the Akana Gateway and ASE, enable sideband authentication by entering the following ASE command:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

- Ensure SSL is configured in ASE for client side connection using CA-signed certificate.Please refer to Configuring SSL for external APIs for more details.
- Generate sideband authentication token by entering the following command in the ASE command-line interface (CLI):

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

- Enable the connection keepalive between gateway and ASE by navigating to `/opt/pingidentity/ase/config/` and setting the value of `enable_sideband_keepalive` to `true` in the `ase.conf` file.
  - If ASE is running, stop it before making the change and start ASE after setting the value. For more information on ASE configuration, see Sideband ASE configuration using the `ase.conf` file.

About this task

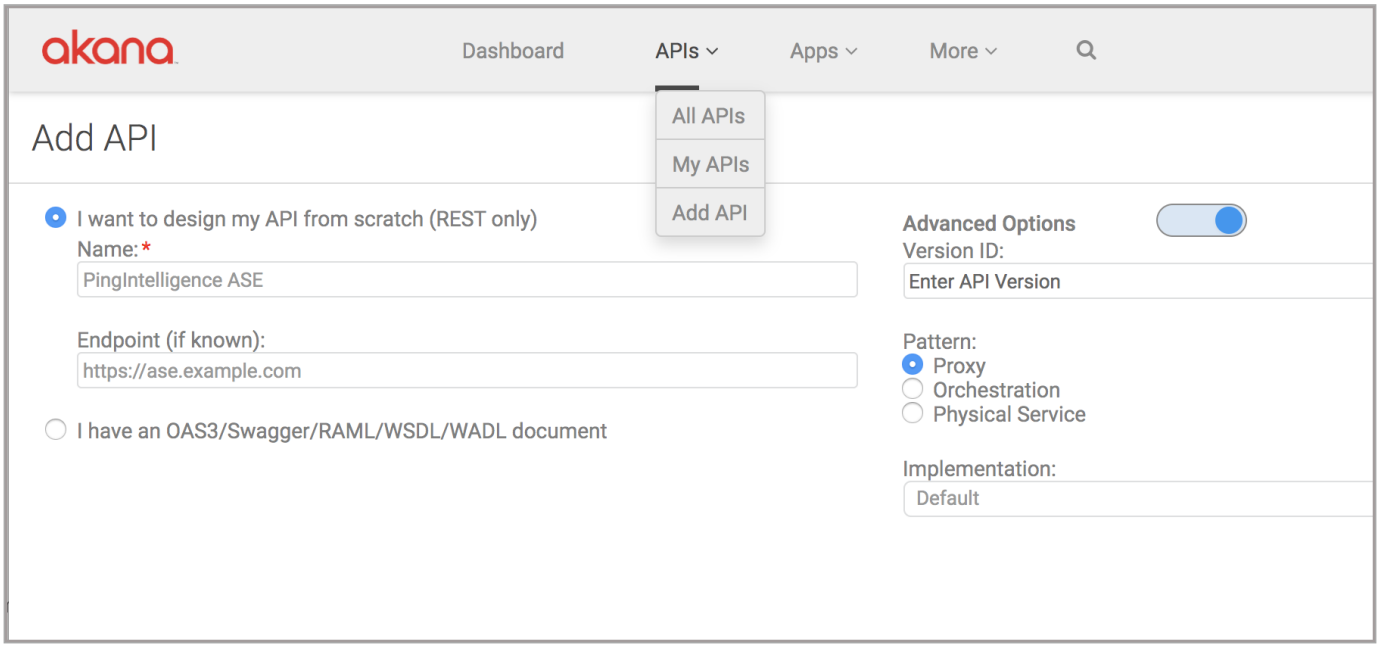
> **Important**
>
> The primary and secondary ASE APIs should not be exposed to external API clients. For more details on securing ASE APIs, see Securing PingIntelligence ASE APIs.

To add ASE APIs to the Akana API Gateway:

Copyright © 2025 Ping Identity Corporation

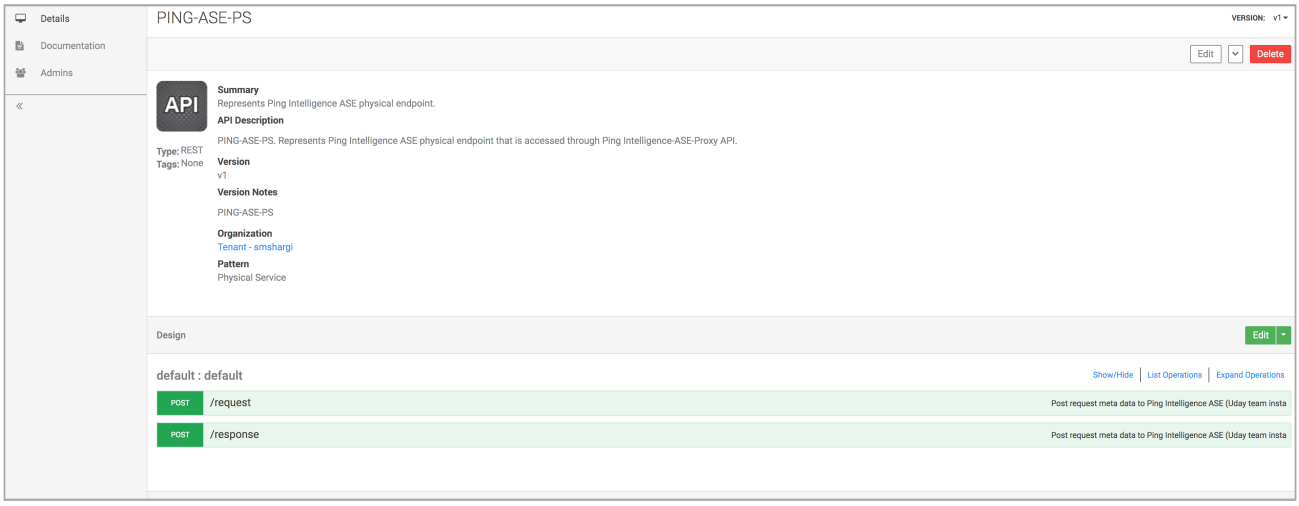
Steps

1. Sign on to the Akana portal and click Add API from the APIs drop-down list.
2. Select I want to design my API from scratch (REST) only.
3. Enter the following details for ASE:
    1. Enter the name of the API in the Name field.
    2. Enter the Endpoint: *<http/https>://<ASE-Hostname or IP>*/ase.
    3. Click the toggle to enable Advanced Options.
    4. Enter the API version in the Version ID field.
    5. For Pattern, select Proxy.
    6. Select Implementation.
    7. Select Deployment Zones.
4. Click Save after entering the details.

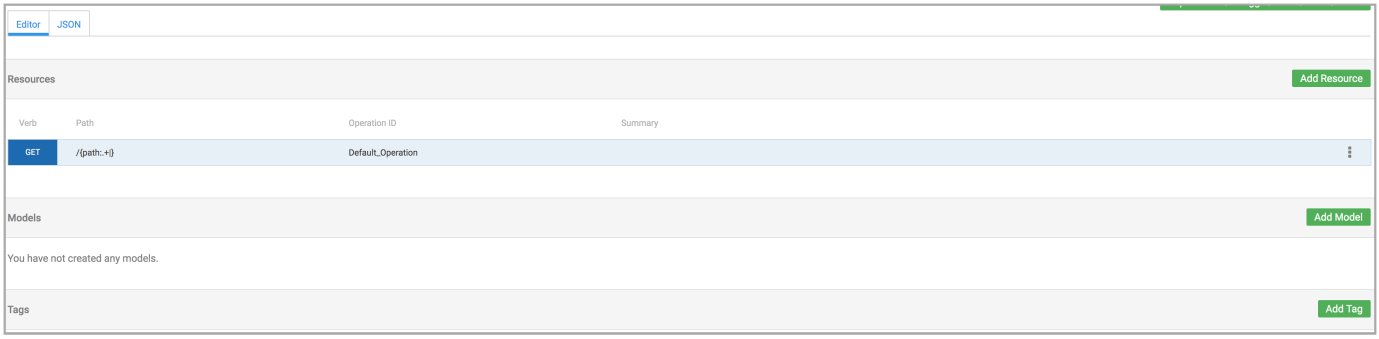


5. Add two resources under Resources: one to post request metadata to ASE and another to post response metadata to ASE.To add a resource to the ASE API, open API Designer:
    1. Navigate to the Overview page of the API.
    2. Choose Details from the left menu pane. The summary of the API is displayed in the details.
    3. In the Design section, click Edit to enter API Designer.

Copyright © 2025 Ping Identity Corporation

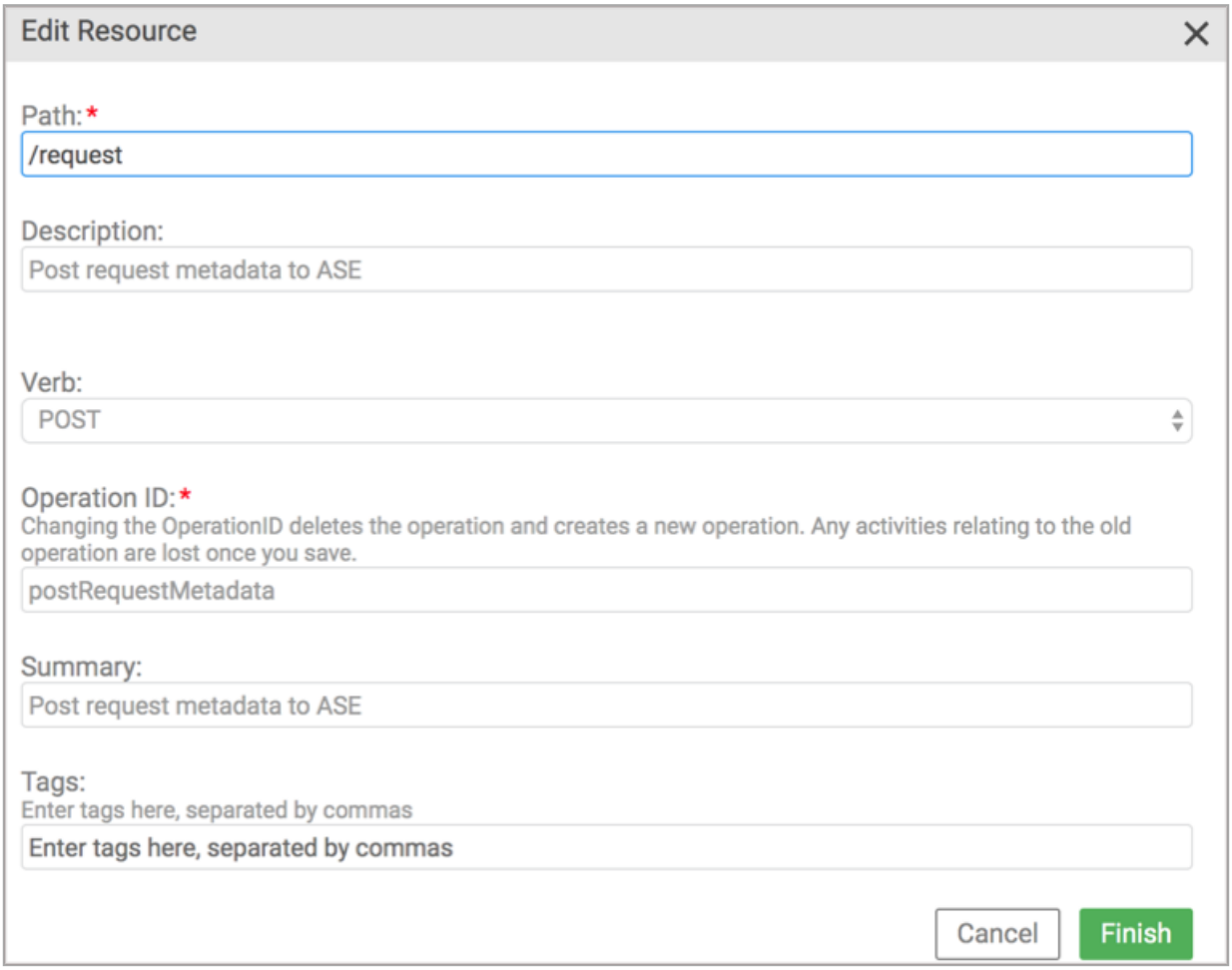


6. Add the Request resource to the API:



1. Click Add Resource to open the Edit Resource window.
2. Enter `/request` in the Path to post request metadata to ASE.
3. For Verb, choose `POST`.
4. Enter Operation ID. If the user does not provide the value, a random value is generated for Operation ID.
5. Click Finish after updating the other optional details like Description, Summary, and Tags.
6. Click Save.

Copyright © 2025 Ping Identity Corporation

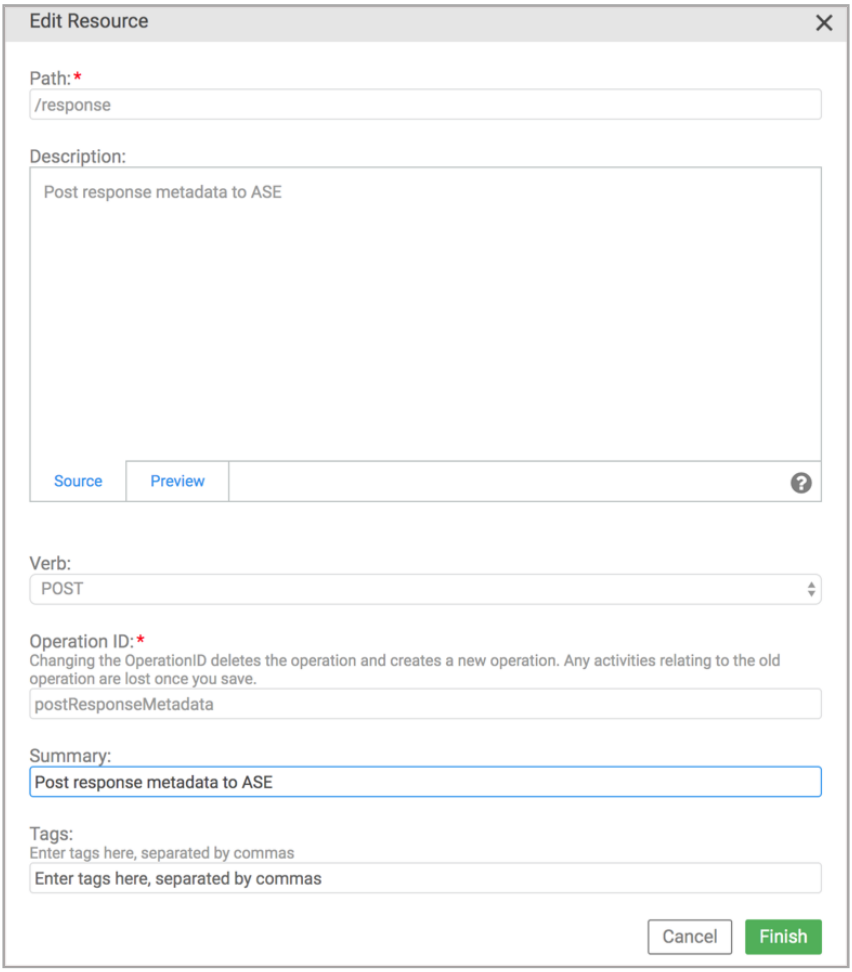


> **Note**
>
> A default resource is created when an API is added to Akana API Gateway. This resource can be edited to add the first resource.

7. To add the Response resource to the API:

1. **Click Add Resource to open the Edit Resource window.**
2. **Enter `/response` in the Path field to post request metadata to ASE.**
3. **For Verb, choose `POST`.**
4. **Enter Operation ID. If the user does not provide the value, a random value is generated for Operation ID.**
5. **Click Finish after updating the other optional details like Description, Summary, and Tags.**

Copyright © 2025 Ping Identity Corporation



8. To add the secondary (backup) ASE node, repeat steps 1- 5.

Securing PingIntelligence ASE APIs

The primary and secondary API Security Enforcer (ASE) APIs added in the Akana API Gateway should be secured from unauthorized access of external clients.

About this task

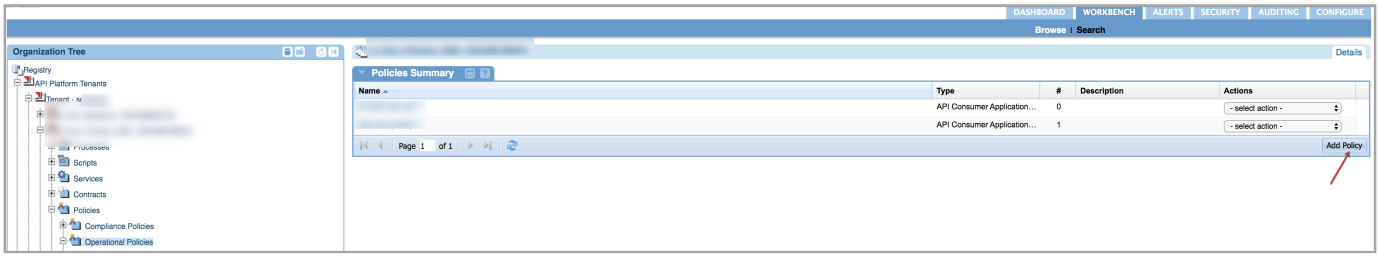
To ensure the Akana API Gateway is secure, you must secure the ASE APIs using the API Consumer Application Security operational policy. The policy allows control on the clients attempting to access the ASE APIs.

To add the API Consumer Application Security operational policy to ASE APIs:

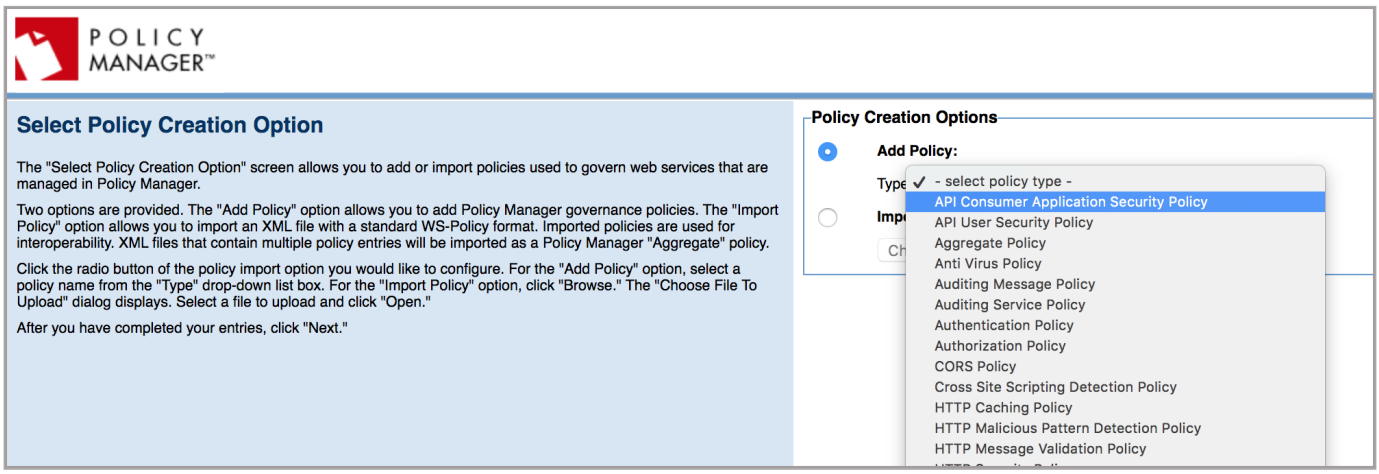
Steps

1. Sign on to Akana Policy Manager, navigate to the Tenant, and select the ASE API.
2. Click + to expand and select Policies. Click Operational Policies and then click Add Policy on the bottom-right.

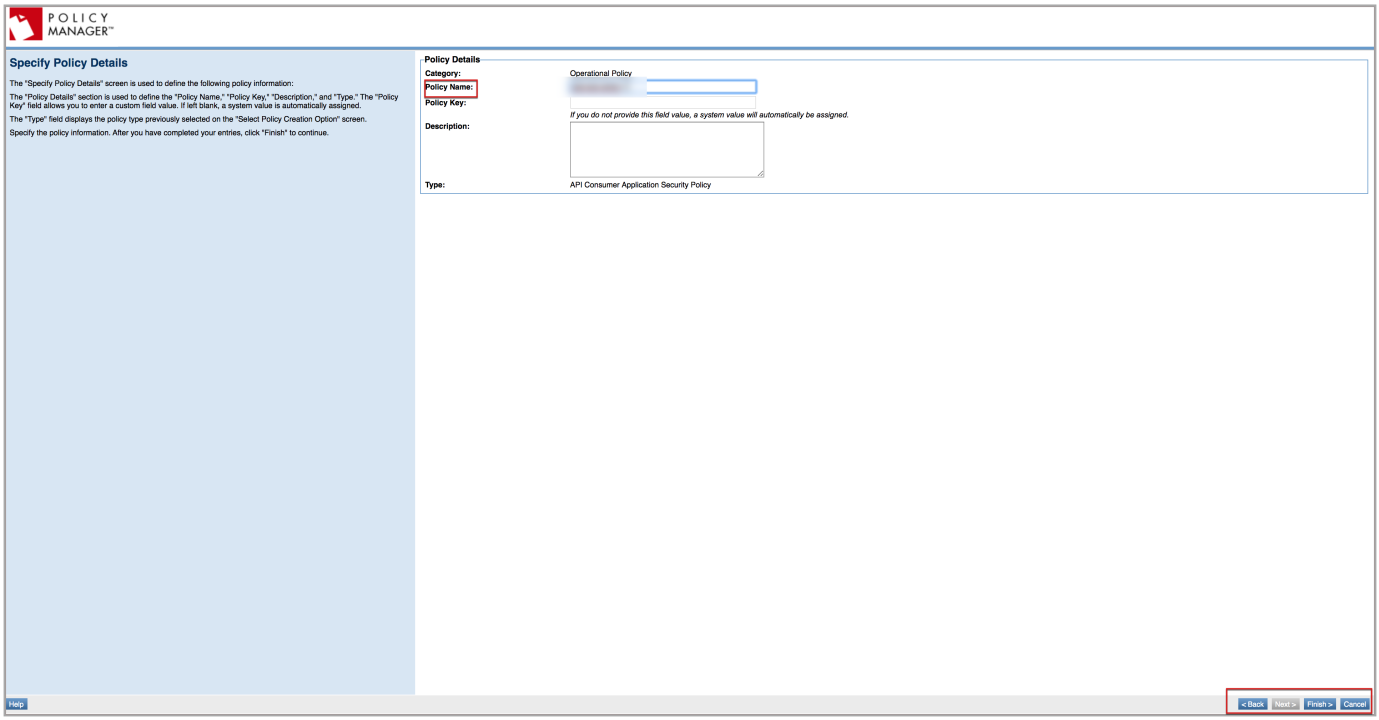
Copyright © 2025 Ping Identity Corporation



3. In the Add Policy wizard, select API Consumer Application Security Policy from the Add Policy drop-down list.



4. Enter a Policy Name, click Next, and then click Finish to save the policy.

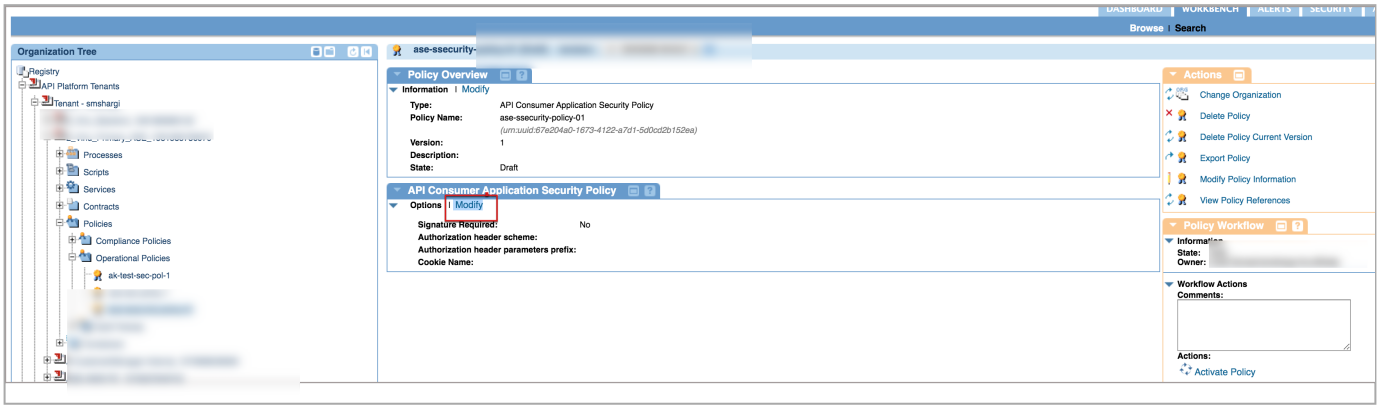


Result:

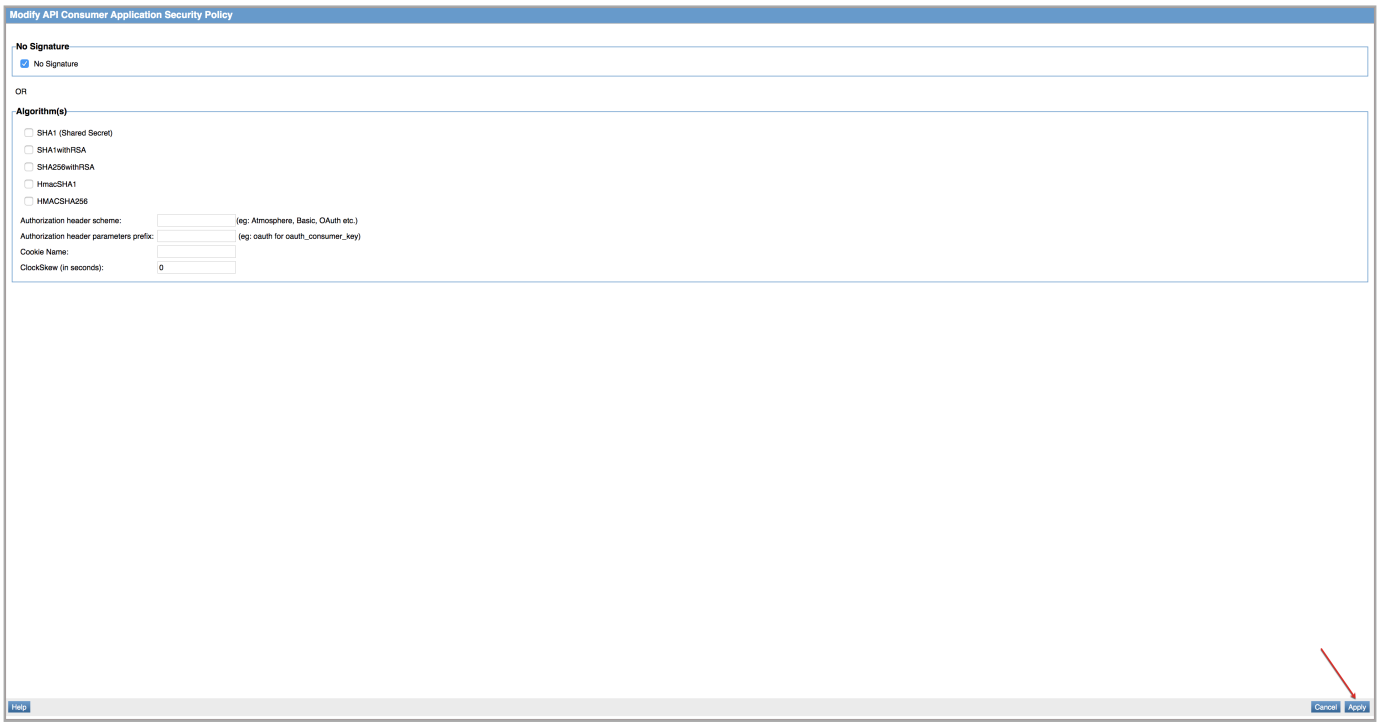
The policy displays under Policies in the ASE API.

5. Click the policy. ClickModify under the API Consumer Application Security Policy section.

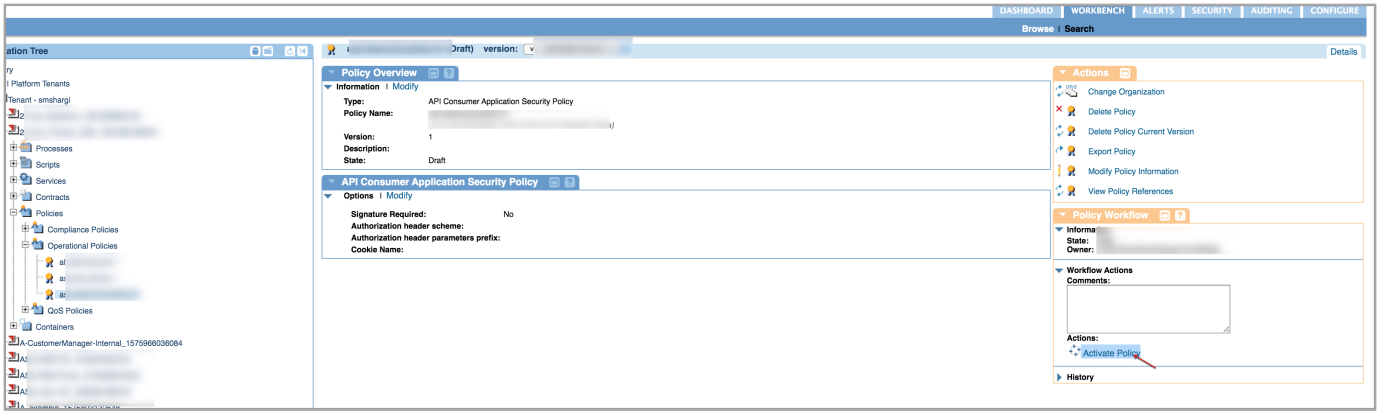
Copyright © 2025 Ping Identity Corporation



6. Click Apply on the Modify API Consumer Application Security Policy page without making any changes.

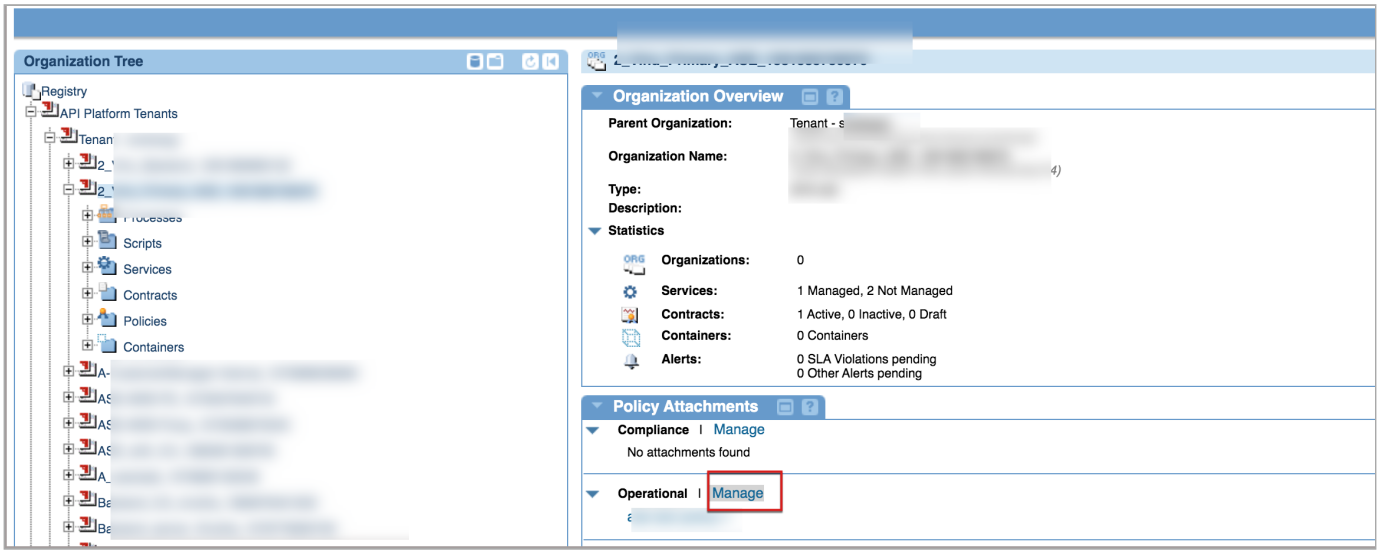


7. Next, click Activate Policy.

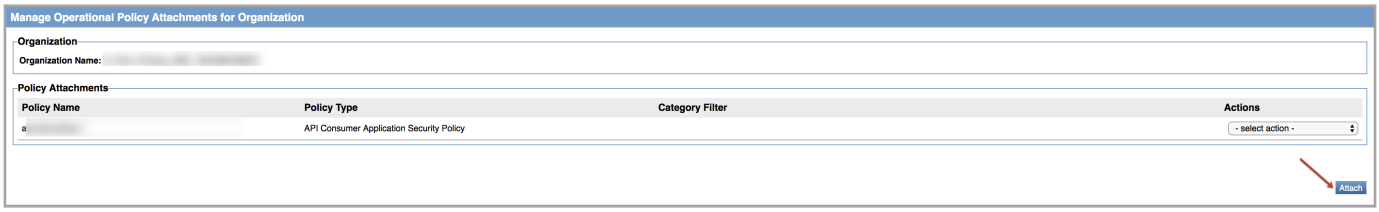


8. Select the ASE API and click Manage in the Policy Attachments section.

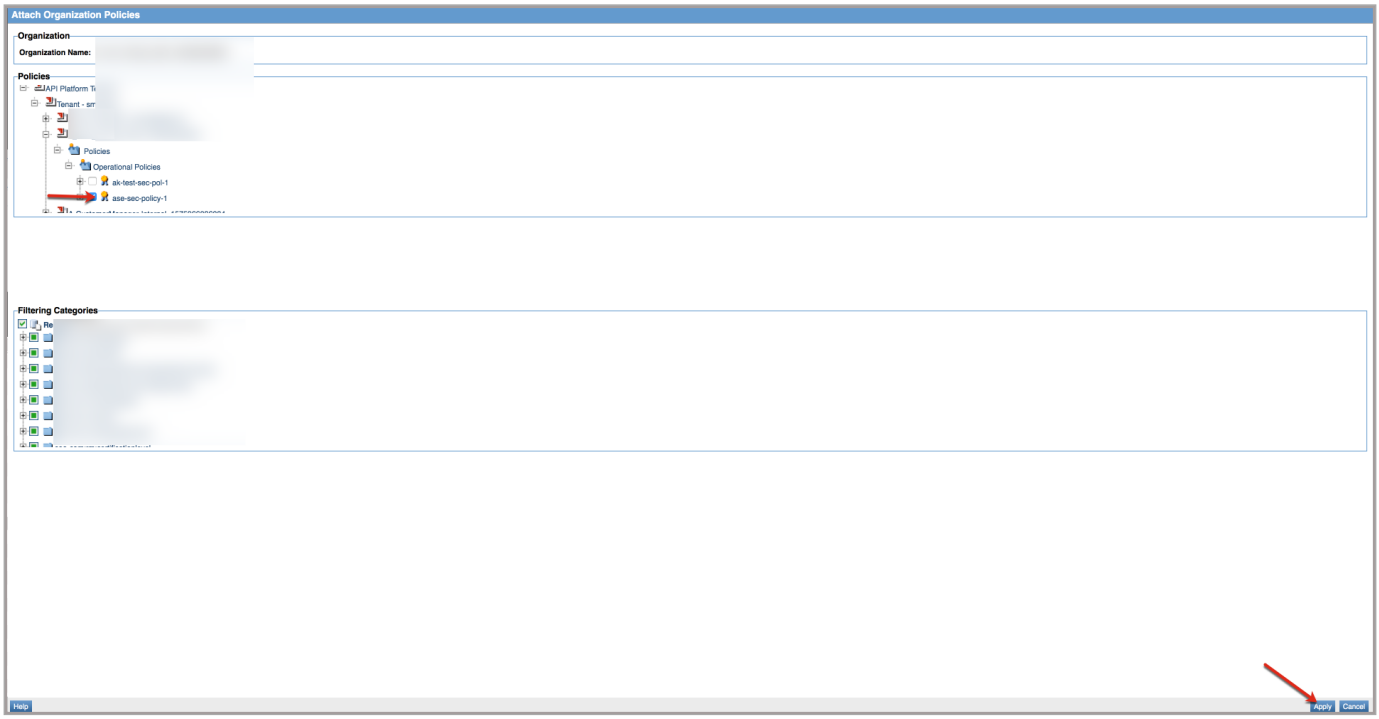
Copyright © 2025 Ping Identity Corporation



9. To attach the policy to ASE API, click Attach on the Manage Operational Policy Attachments for Organization page.



10. In the Attach Organization Policies window, select the policy added from the Policies window, and select the check box. For the policy click Apply.



11. Click Close.

12. To add the policy to a secondary ASE API, repeat steps 1-11.

Copyright © 2025 Ping Identity Corporation

Capturing ASE details

Capture the Service QName, Interface Name, and Operation Name for the primary and secondary ASE nodes.

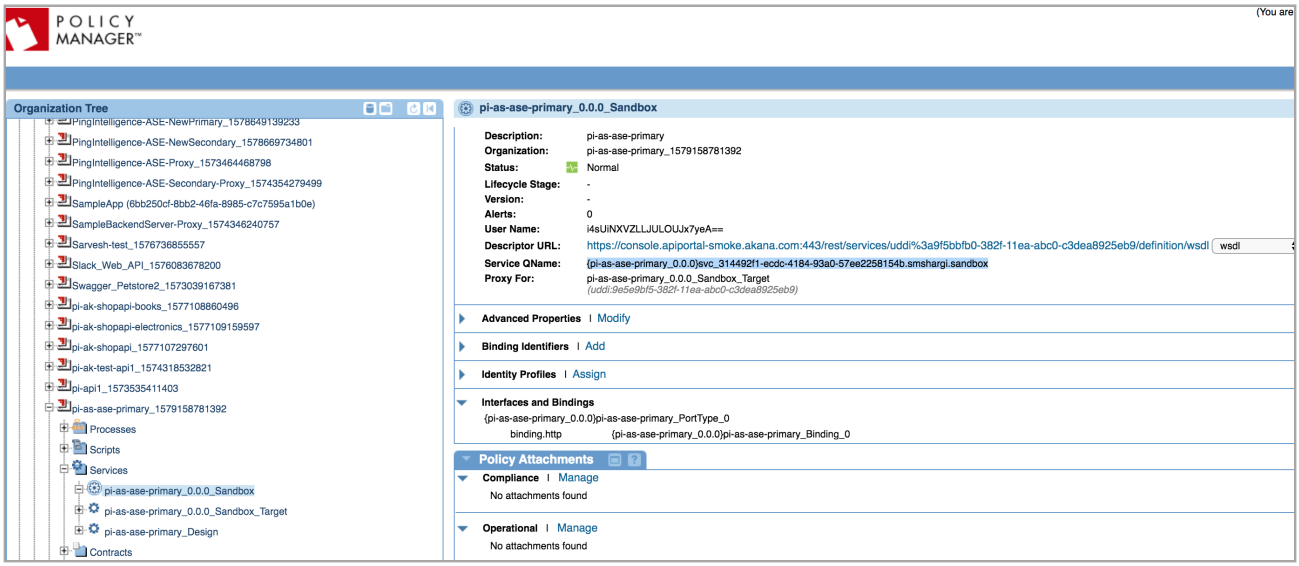
About this task

The Service QName, Interface Name, and Operation Name are used in `config.js`.

To capture these values:

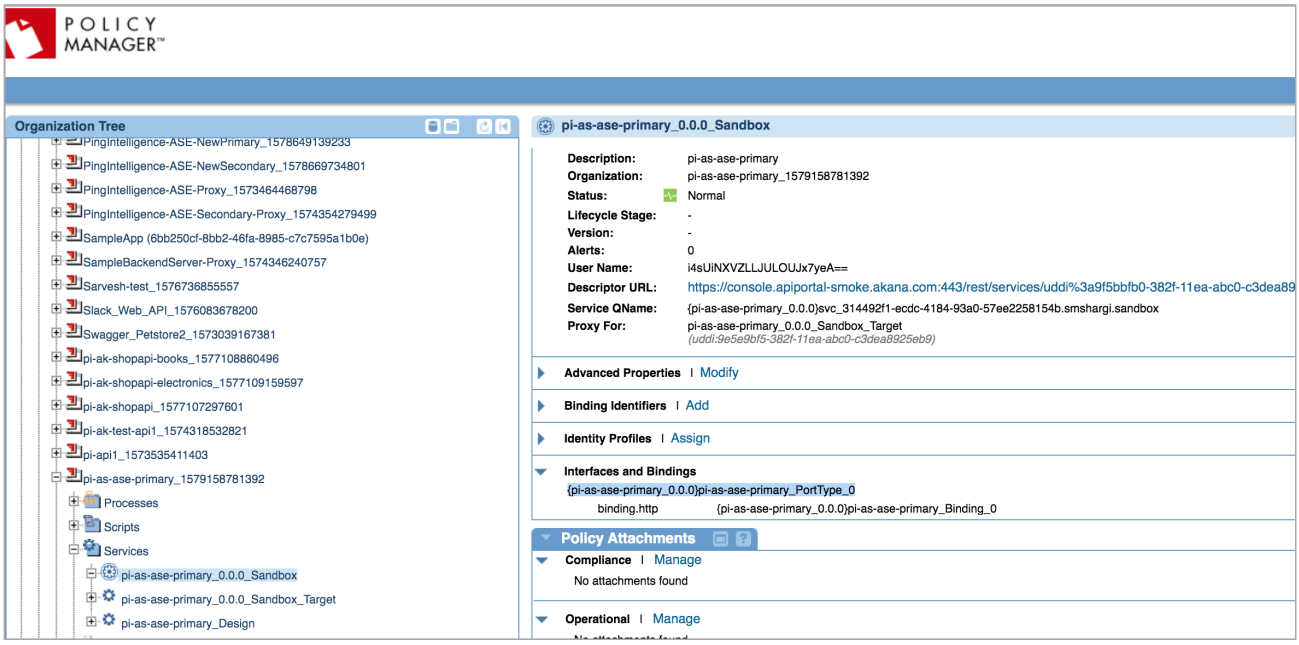
Steps

1. Sign on to Akana Policy Manager, navigate to the Organization Tree on the left, and select the Tenant and then the ASE API.
2. Expand the Services and click on the API:
    1. Copy the Service QName and paste it into a text editor.

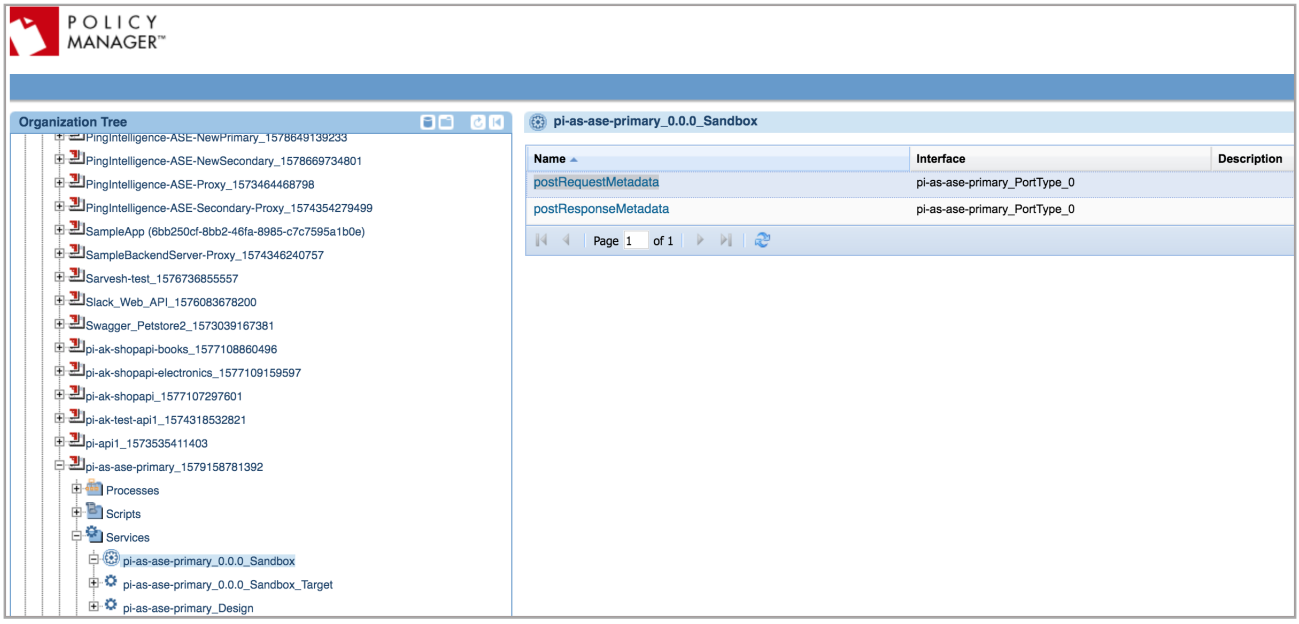


    2. Under Interfaces and Bindings, copy the Interface Name and paste it into a text editor.

Copyright © 2025 Ping Identity Corporation



3. Click Operations tab on the menu, copy Operation Name, and paste it into a text editor.



3. To capture the Service Qname, Interface Name, and Operation Name details for a secondary ASE API, repeat steps 1-2.

Next steps

Use the captured values to deploy a policy in the Akana API Gateway. See Deploying PingIntelligence policies.

Deploying PingIntelligence policies

Deploying PingIntelligence policies in the Akana API Gateway is divided into three parts:

Copyright © 2025 Ping Identity Corporation

About this task

- Adding an input script (`config.js`)
- Adding a PingIntelligence policy and applying the policy to APIs
- Adding the RetainerHeader policy and applying the policy to APIs

Downloading PingIntelligence policies

About this task

To download and extract the PingIntelligence policy:

Steps

1. Download the PingIntelligence policy.
2. Extract the policies by running the following command:

```
# tar -zxvf  <file name>
```

Example:

```
# tar -zxvf pi-api-akana-policy-4.1.1.tar.gz
```

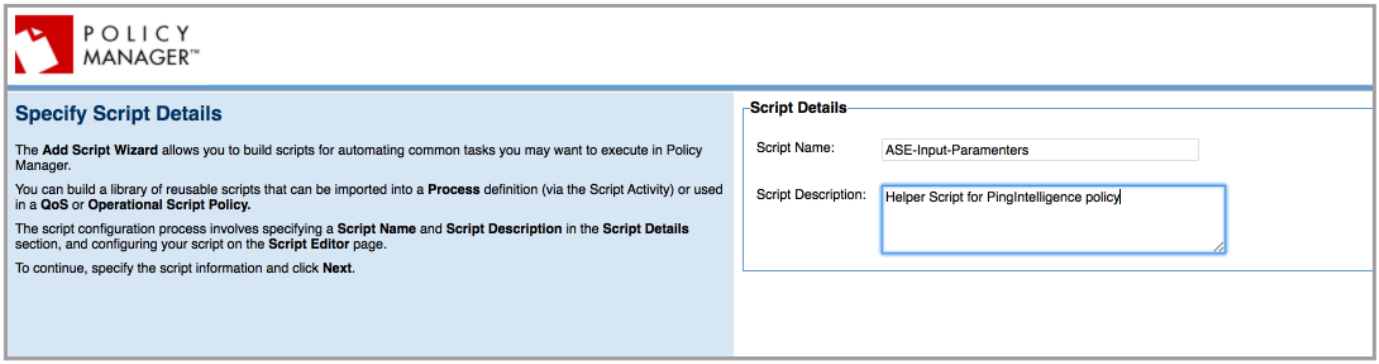
Adding an input script

About this task

To add an input script to the Akana API gateway:

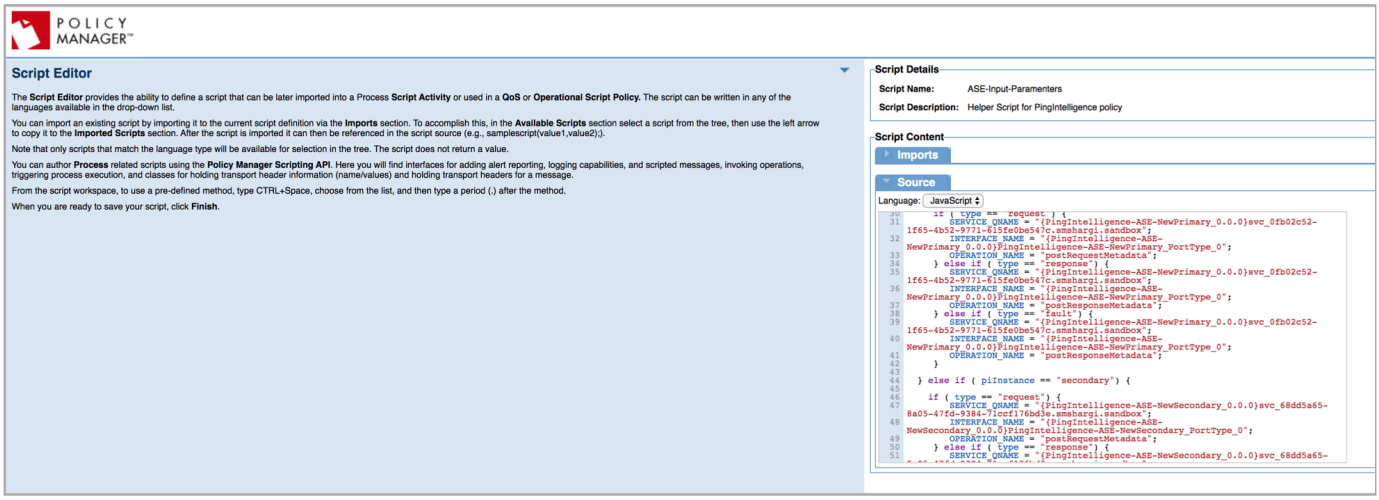
Steps

1. Sign on to Akana Policy Manager, navigate to Tenant, and click Scripts.
2. Click Add Script.
3. Enter Script Name and Script Description, and clickNext.



4. Select JavaScript for Language from the list.
5. Copy the contents of the `config.js` script provided by PingIntelligence and paste them into the Source.

Copyright © 2025 Ping Identity Corporation



6. Paste the values of Service_QName, Interface_Name, and Operation_Name that were copied in Capturing ASE details.

   This needs to be done for both primary and secondary ASE nodes. The following table lists the variables in `config.js` that need to be populated.

Variable	Description
ase_token	Variable for the ASE sideband authentication token.
primary_ase_service	Service QName for primary ASE.
primary_ase_interface	Interface Name for primary ASE.
primary_ase_request_operation	Operation Name for posting Request Metadata in primary ASE.
primary_ase_response_operation	Operation Name for posting Response Metadata in primary ASE.
secondry_ase_service	Service QName for secondary ASE.
secondary_ase_interface	Interface Name for secondary ASE.
secondary_ase_request_operation	Operation Name for posting Request Metadata in secondary ASE.
secondary_ase_response_operation	Operation Name for posting Response Metadata in secondary ASE.

   Example:

   Below is a sample substitution snippet:

Copyright © 2025 Ping Identity Corporation

```
var ase_token = "ASE-Token-123";
/Primary ASE Configuration/
var primary_ase_service = "{pi-as-ase-primary_0.0.0}svc_314492f1-
ecdc-4184-93a0-57ee2258154b.smshargi.sandbox";
var primary_ase_interface = "{pi-as-ase-primary_0.0.0}pi-as-ase-primary_PortType_0";
var primary_ase_request_operation = "postRequestMetadata";
var primary_ase_response_operation = "postResponseMetadata";
/**/
/Secondary ASE Configuration/
var secondry_ase_service = "{pi-as-ase-primary_0.0.0}svc_314492f1-
ecdc-4184-93a0-57ee2258154b.smshargi.sandbox";
var secondary_ase_interface = "{pi-as-ase-primary_0.0.0}pi-as-ase-primary_PortType_0";
var secondary_ase_request_operation = "postRequestMetadata";
var secondary_ase_response_operation = "postResponseMetadata";
```

7. Click Finish, and then click Close.

Adding a PingIntelligence policy

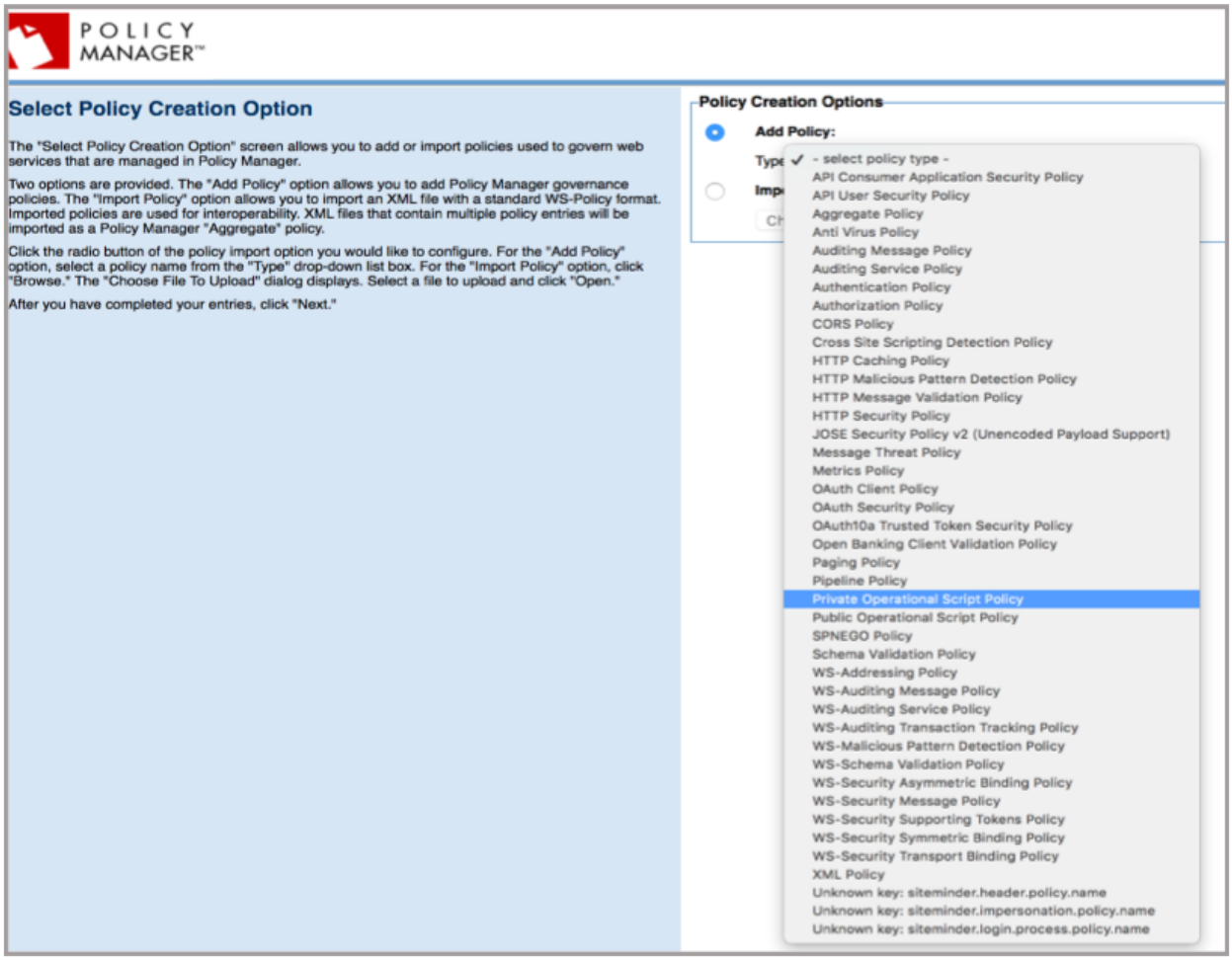
About this task

To add a PingIntelligence policy to the Akana API Gateway:

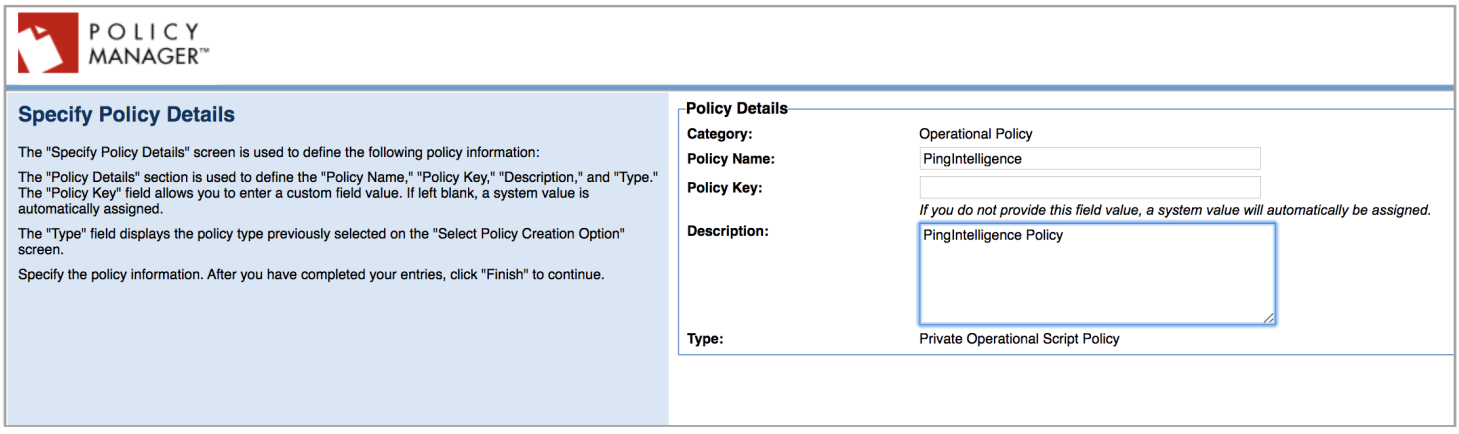
Steps

1. Sign on to Akana Policy Manager, navigate to the Tenant, and click Operational Policies under Policies.
2. Select Add Policy.
3. For Type, select Private Operational Script Policy from the drop-down list, and click Next.

Copyright © 2025 Ping Identity Corporation

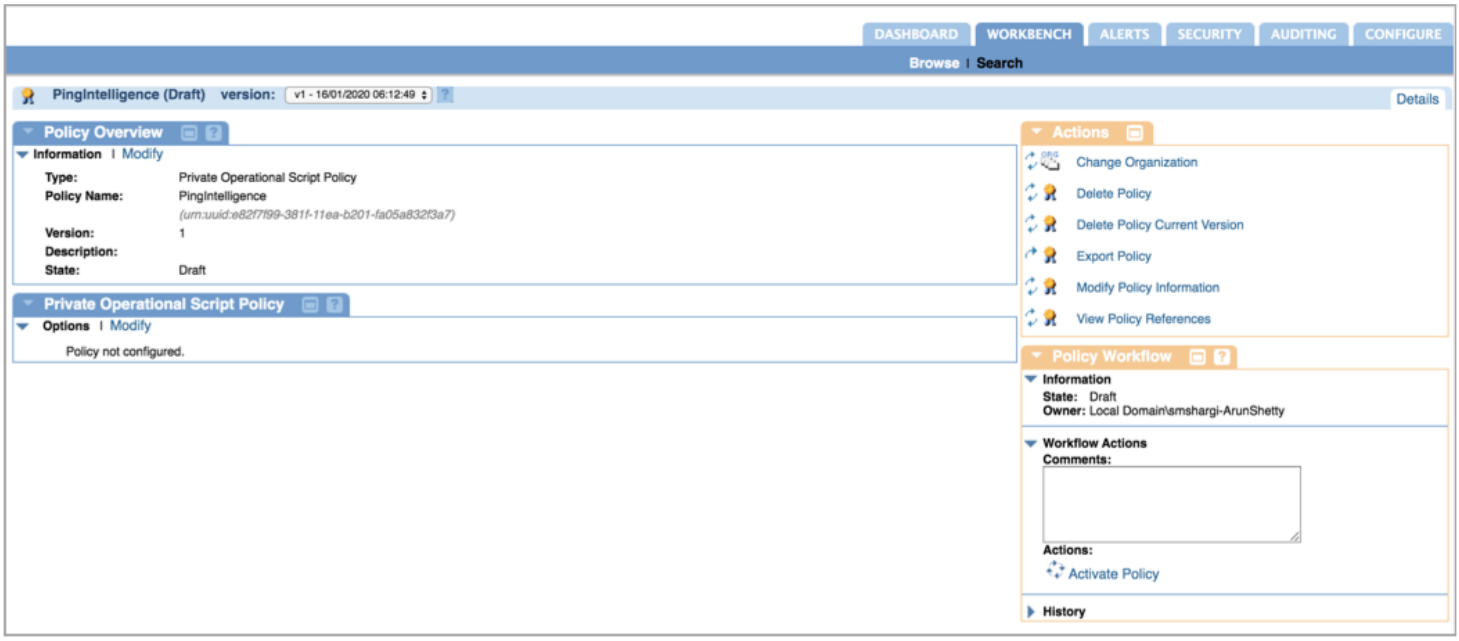


1. Enter Policy Name and Description, click Finish, and then click Close.

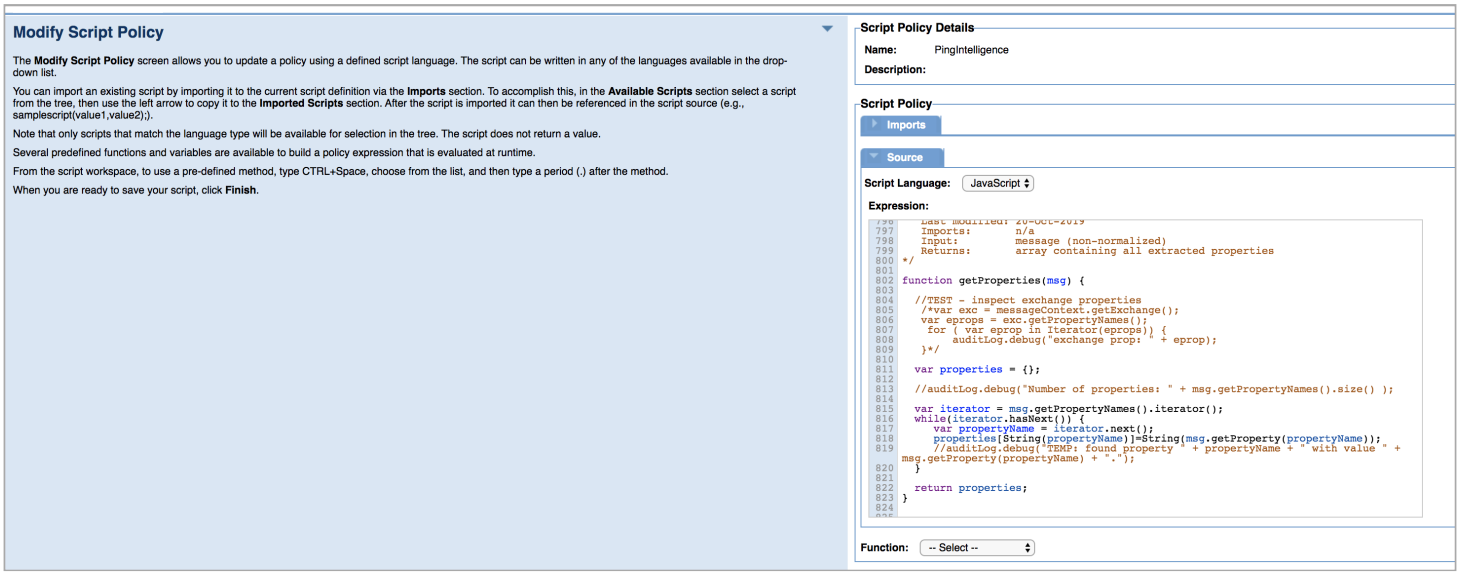


1. Navigate to Workbench.
2. In the Private Operational Script Policy section, select the policy name and click Modify.

Copyright © 2025 Ping Identity Corporation



1. Click on Imports. Select the script added in Adding an input script and import it by clicking <<.
2. Select JavaScript for Language from the list.
3. Copy the contents of the `pi_policy.js` script and paste them into Expression under Source.



1. Click Finish and then click Close.
2. In the WorkFlow Actions, click Activate Policy to activate the PingIntelligence policy.

Applying the PingIntelligence policy to APIs

About this task

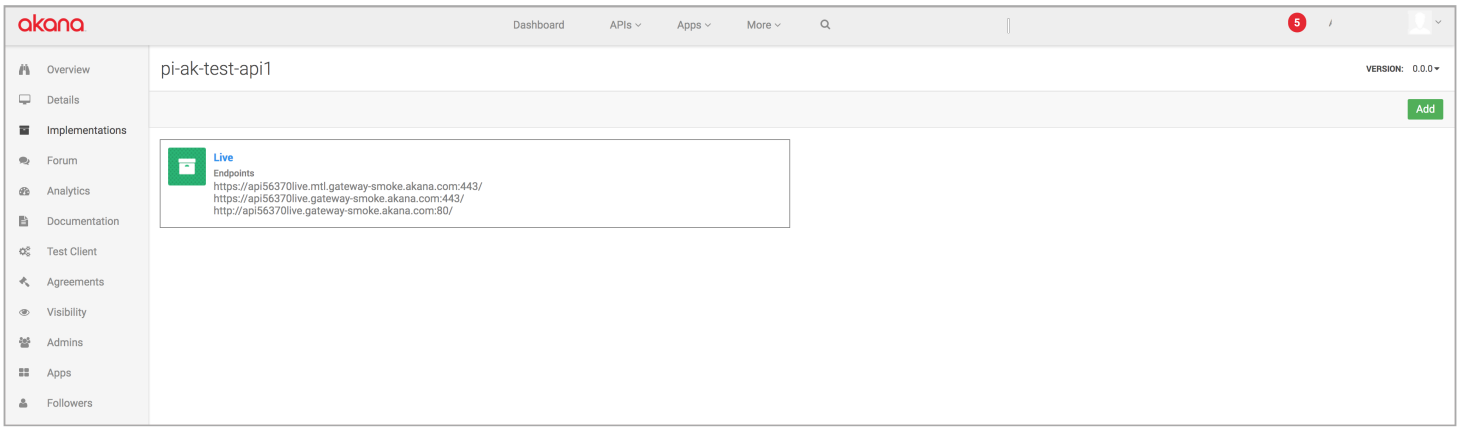
The PingIntelligence policy can be applied at tenant level, org level and at individual API level.

To add a policy at the API level:

Copyright © 2025 Ping Identity Corporation

Steps

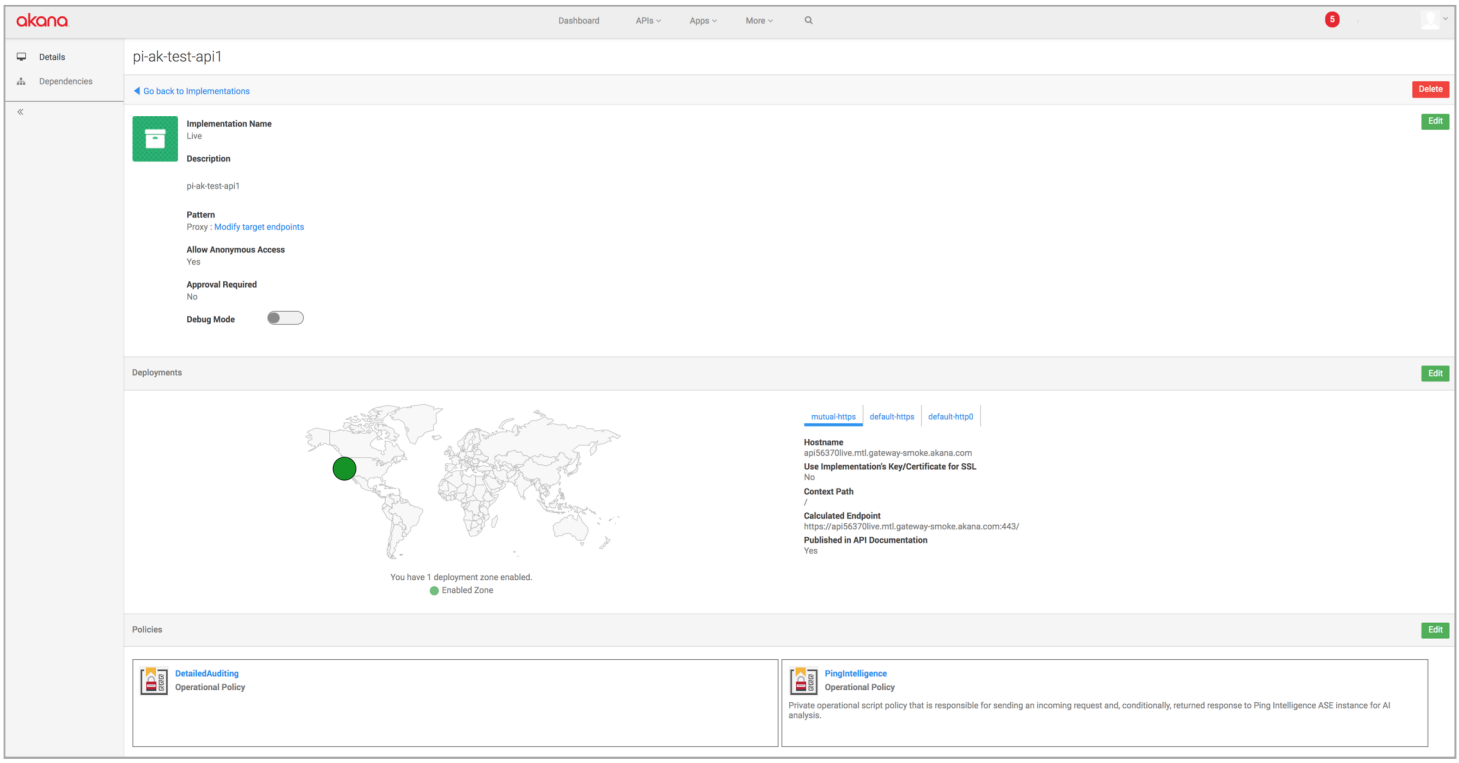
1. Sign on to Akana Portal.
2. Click the API name.
3. In the left navigation, click Implementations.



1. Click the API Implementation Name icon.

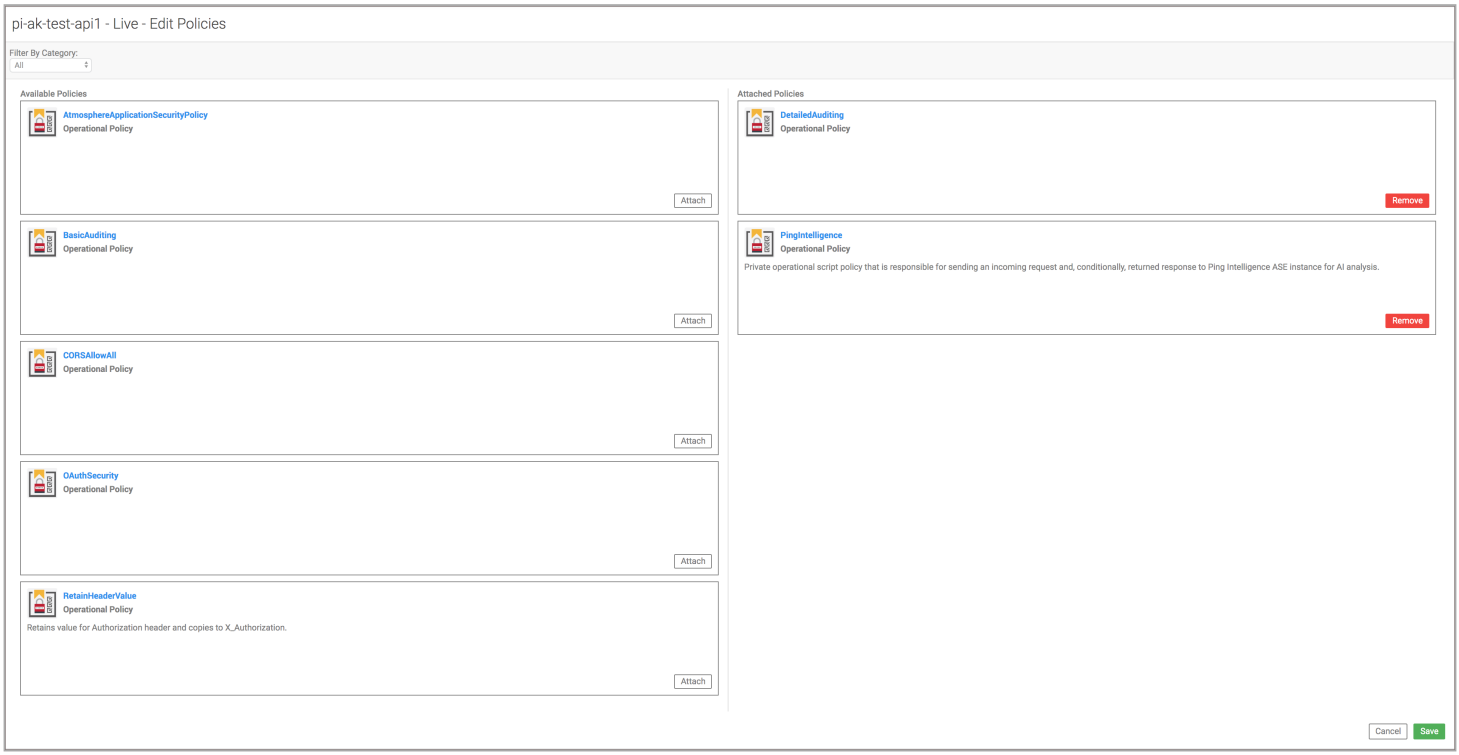
   Possible values for API Implementation are `Live`, `Sandbox`, or `Development`.

2. In the Policies section, click Edit.



1. Find the PingIntelligence policy in the Available Policies pane, and click Attach under the PingIntelligence policy.
2. Click Save.

Copyright © 2025 Ping Identity Corporation



Adding the RetainHeader policy

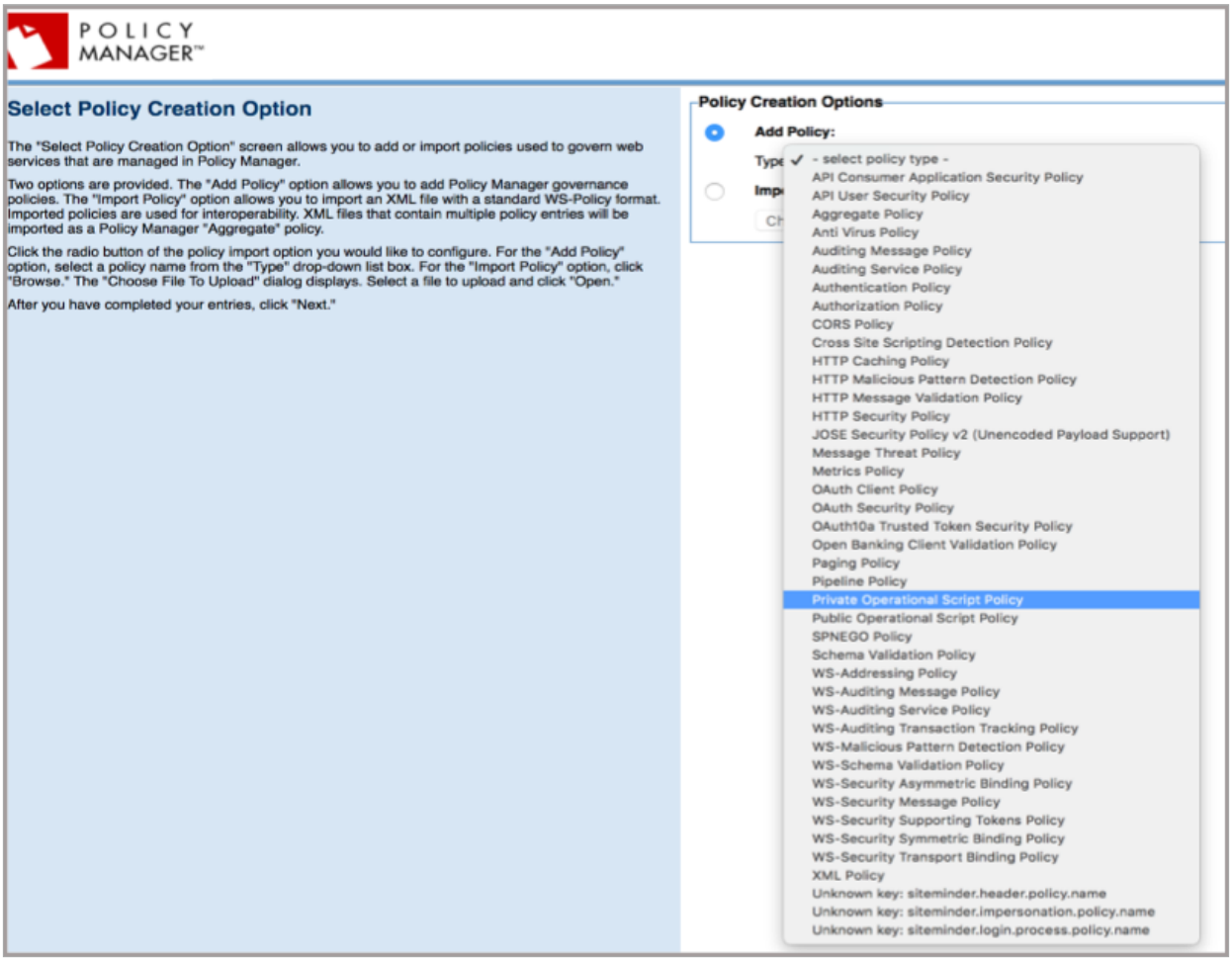
About this task

To add the RetainHeader policy to the Akana API Gateway:

Steps

1. Sign on to Akana Policy Manager and click Tenant in the left navigation.
2. Under Policies, click Operational Policies.
3. Select Add Policy and choose Private Operational Script Policy from the Type drop-down list.

Copyright © 2025 Ping Identity Corporation



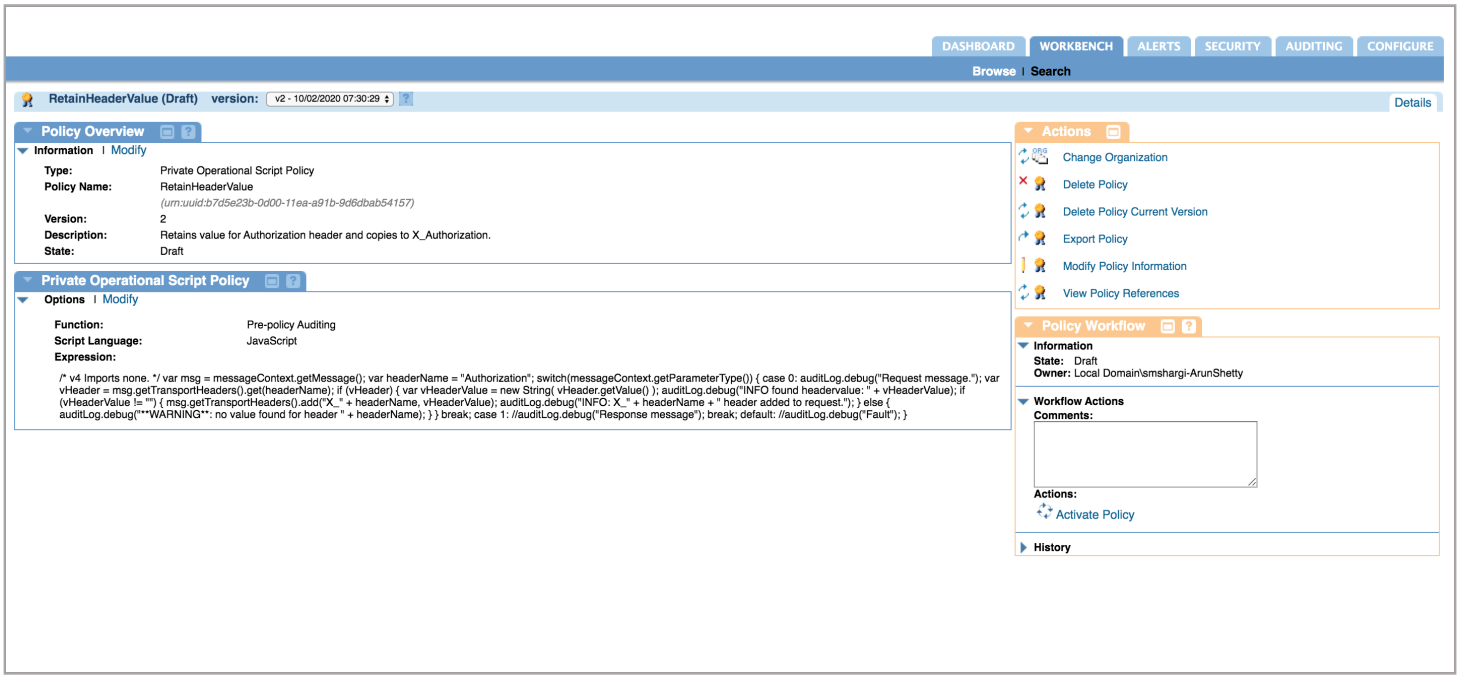
4. Click Next.

5. Enter Policy Name and Description, click Finish, and then click Close.

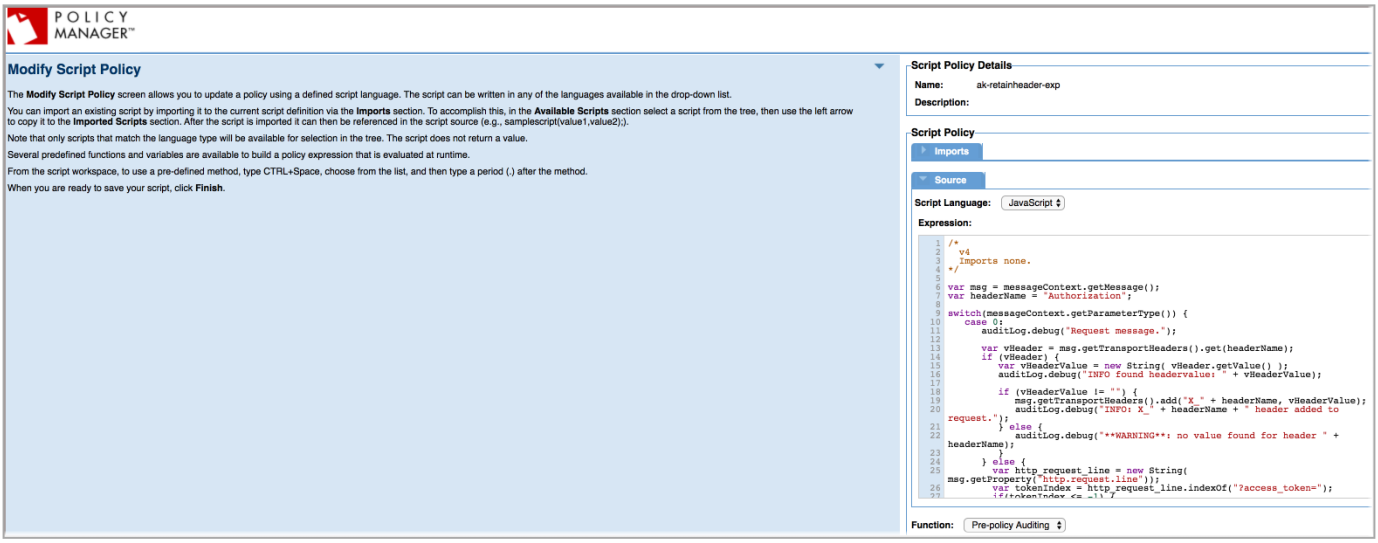
1. Navigate to the Workbench tab.

2. In the Private Operational Script Policy section, select the policy name and click Modify.

Copyright © 2025 Ping Identity Corporation



1. For Script Language, select JavaScript from the drop-down list.
2. Copy the contents of the `retain-header-policy.js` script and paste them into Expression.
3. For Function, select Pre-policy Auditing from the drop-down list.



4. Click Finish and then click Close.
5. Under the WorkFlow Actions, click Activate Policy to activate the RetainHeader policy.

Applying the RetainHeader policy to APIs

About this task

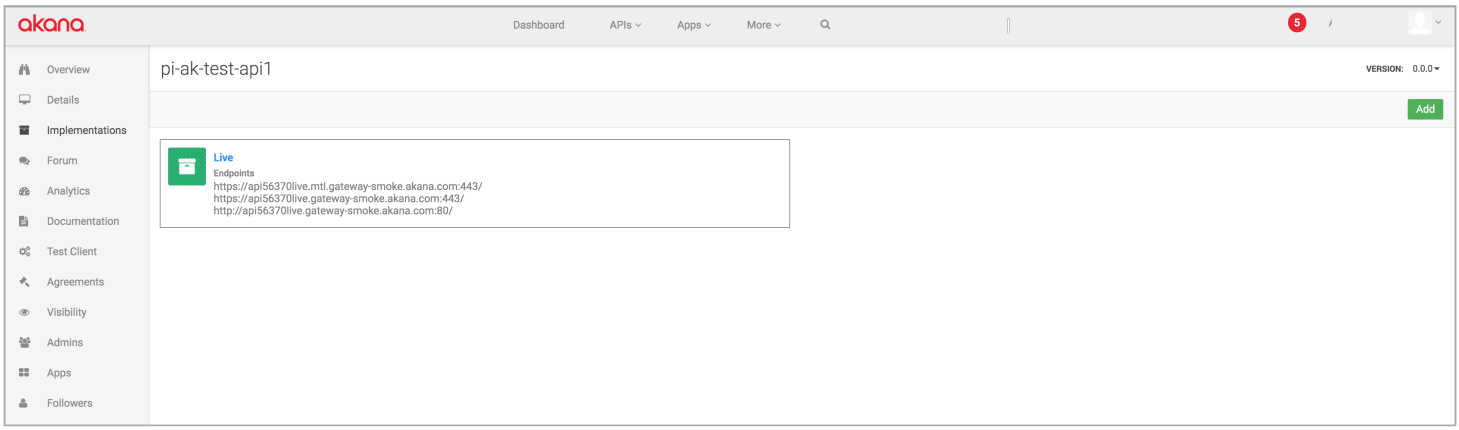
The RetainHeader policy can be applied at the tenant, org, and at individual API level.

To add a policy at the API level:

Copyright © 2025 Ping Identity Corporation

Steps

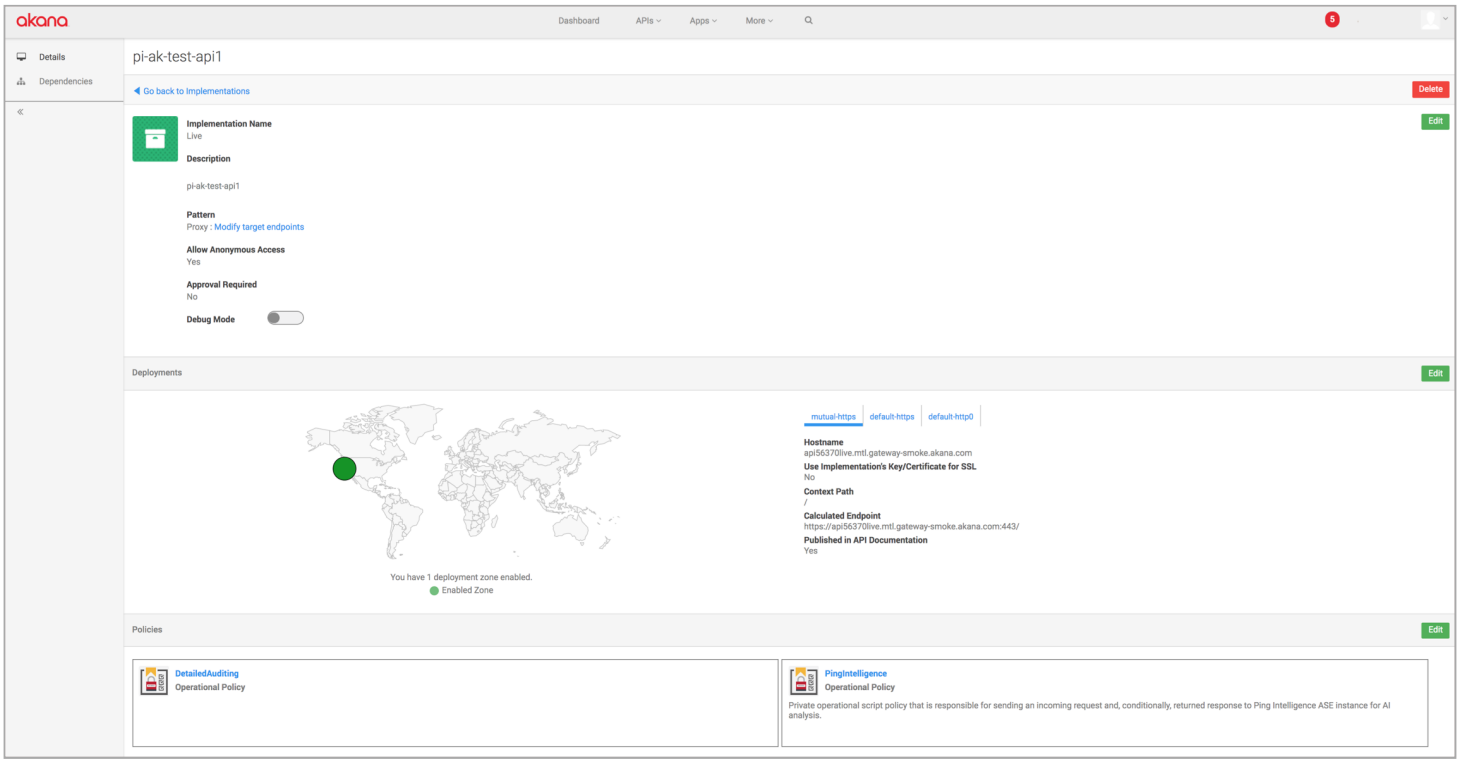
1. Sign on to the Akana Portal.
2. Click the API name.
3. In the left navigation, click Implementations.



1. Click the API Implementation Name icon.

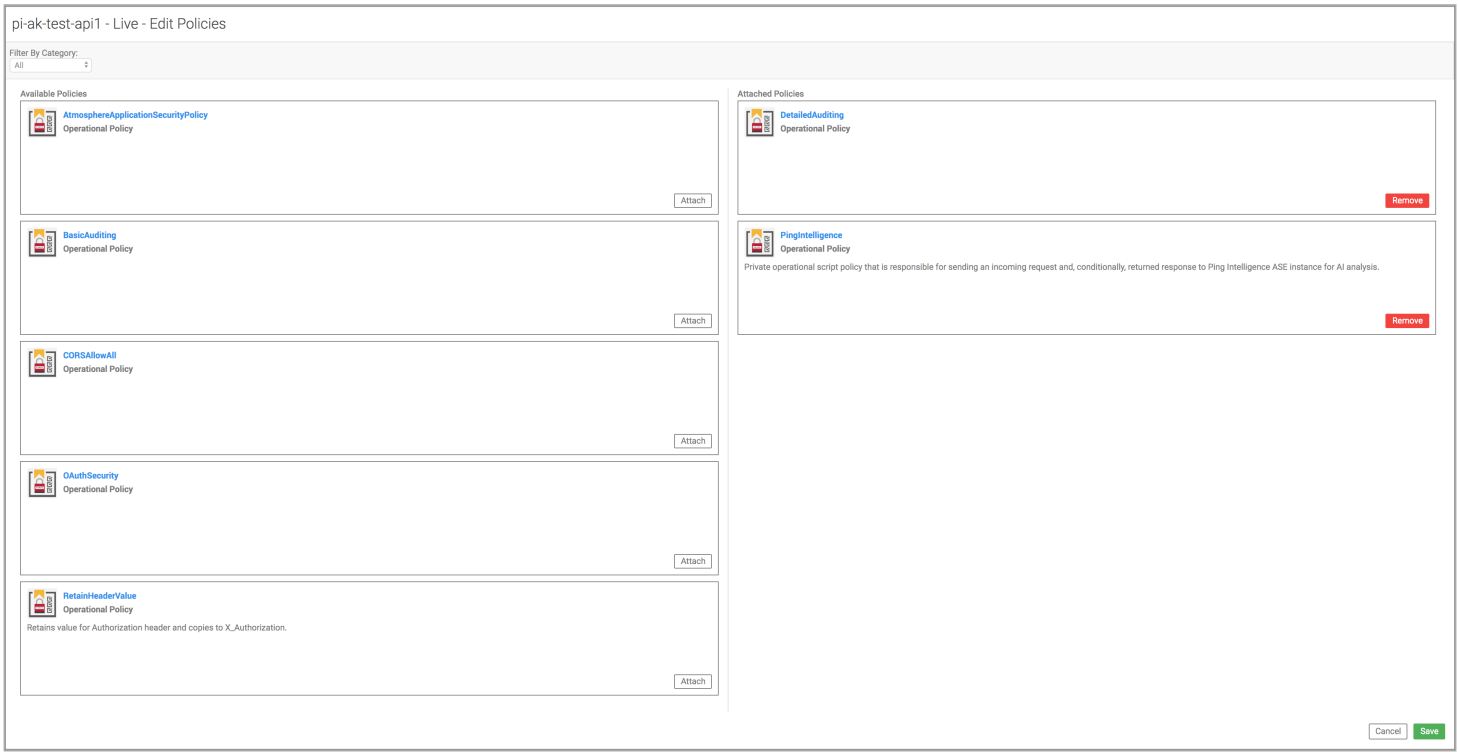
   Possible values for API Implementation are `Live`, `Sandbox`, or `Development`.

2. In the Policies section, click Edit.



1. Under Available Policies, find the RetainHeader policy, and click Attach under the RetainHeader policy.
2. Click Save.

Copyright © 2025 Ping Identity Corporation



Apigee integration

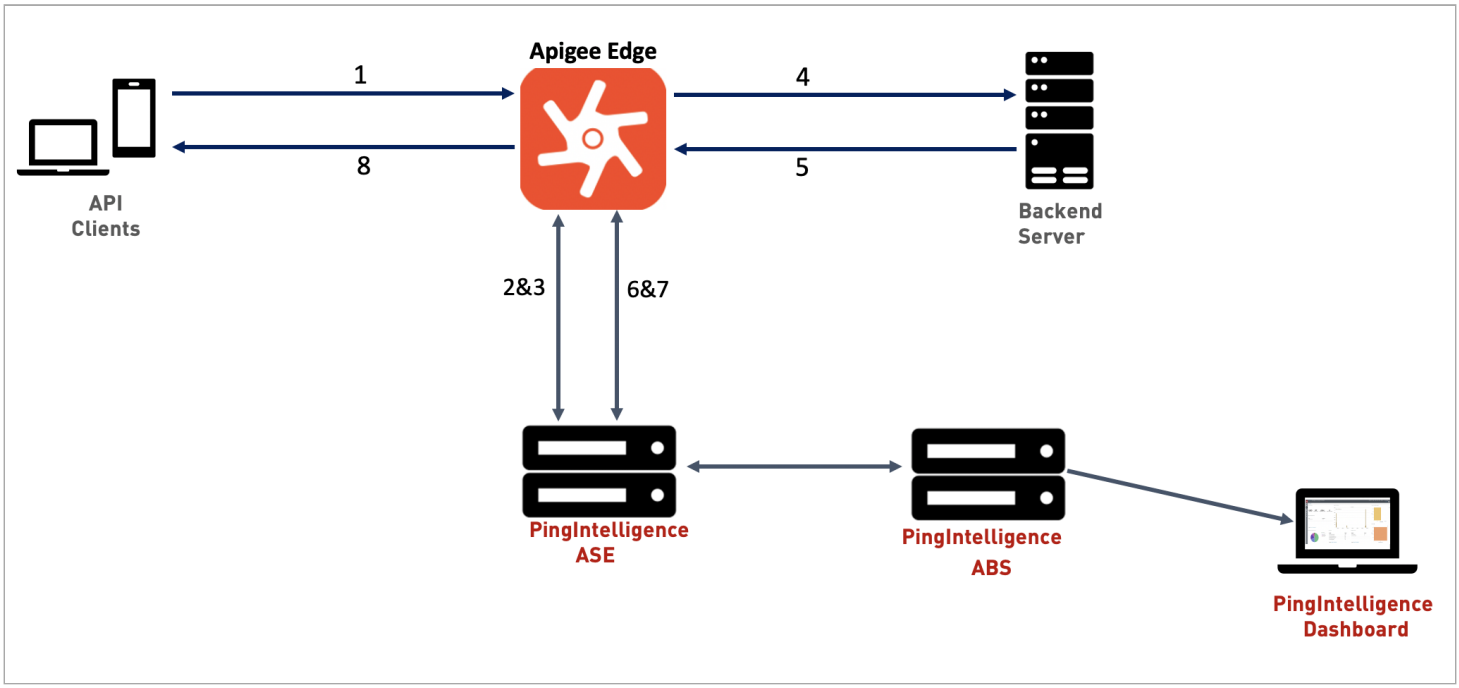
PingIntelligence provides a shared flow to integrate Apigee Edge with PingIntelligence for APIs platform.

The two mechanisms of calling shared flows are flow hook and flow callout policies. A flow hook in Apigee Edge applies the PingIntelligence shared flow globally to all APIs in an environment in an organization. The FlowCallout policy in Apigee Edge applies the PingIntelligence shared flow on a per API basis in an environment in an organization.

PingIntelligence provides an automated tool to deploy both flow hook and flow callout polices.

The following diagram shows the logical setup of PingIntelligence API Security Enforcer (ASE) and Apigee Edge.

Copyright © 2025 Ping Identity Corporation



Traffic flows through the Apigee Edge andPingIntelligence for APIs components as follows:

1. Incoming request to Apigee Edge from a client.
2. Apigee Edge makes an API call to send the request information to ASE.
3. ASE checks the request against a registered set of APIs and checks the origin Internet Protocol (IP), cookie, OAuth2 token, or API key against the deny list. If all checks pass, ASE returns a `200-OK` response to the Apigee Edge. If not, a different response code (403) is sent to Apigee Edge. The request information is also logged by ASE and sent to the ABS artificial intelligence (AI) engine for processing.
4. If Apigee Edge receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, the gateway optionally blocks the client. In synchronous mode, the gateway waits for a response from ASE before forwarding the request to backend server. However, if asynchronous mode is enabled, the gateway forwards the request to the backend server without waiting for the response from ASE. The ASE passively logs the request and forwards it to ABS for attack analysis. It performs attack detection without blocking of attacks.
5. Apigee Edge receives the response from the backend server.
6. Apigee Edge makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to Apigee Edge.
8. Apigee Edge sends the response received from the backend server to the client.

Preparing to deploy the PingIntelligence shared flow

Confirm that the following prerequisites are met before using the PingIntelligence Apigee tool.

About this task

Copyright © 2025 Ping Identity Corporation

Before using the PingIntelligence Apigee tool:

Steps

1. Confirm Apigee version.

   PingIntelligence supports Apigee API gateways supporting shared flows.

2. Confirm one of the OpenJDK versions between 11.0.2 to 11.0.6 is on the machine where the PingIntelligence Apigee deployment tool is installed.

3. Install and configure PingIntelligence software 4.0 or higher.

   For information on installing PingIntelligencesoftware, see PingIntelligence automated deployment for virtual machines and servers.

4. Verify API Security Enforcer (ASE) is in `sideband` mode by running the following command in the ASE command line:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

   ```
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : enabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
   ```

   Troubleshooting:

   If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

5. For a secure communication between Apigee Edge and ASE, enable sideband authentication by entering the following command in the ASE command line:

   ```
   # ./bin/cli.sh enable_sideband_authentication -u admin –p
   ```

6. To generate a sideband authentication token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use.

   A token is required for Apigee Edge to authenticate with ASE. This token is generated in ASE and configured in the `apigee.properties` file of the PingIntelligence automated policy tool.

   ```
   # ./bin/cli.sh -u admin -p admin create_sideband_token
   ```


Copyright © 2025 Ping Identity Corporation

7. Verify the certificate in `ase.pem` when using self-signed certificates.
    1. Make sure that the certificate applied for the ASE data port matches with the certificate present in the `ase.pem` certificate file to prevent SSL issues after policy deployment.
    2. Run the following command to obtain the certificate used in the ASE data port. If the certificates do not match, paste the correct certificate in the `/opt/pingidentity/pi/apigee/certs/ase.pem` file.

```
# openssl s_client -showcerts -connect  <ASE IP address>:<port no>  </dev/null 2>/dev/null |
openssl x509 -outform PEM > ase.pem
```

Downloading and installing the automated policy tool

The automated policy tool deploys both flow hook and flow callout polices.

About this task

To download and install the PingIntelligence policy tool:

Steps

1. Download the PingIntelligence policy tool to the `/opt` directory.
2. Untar the policy tool:
    1. At the command prompt, run the following command:

```
tar –zxvf  <filename>
```

Example:

The following example shows what this command could look like in your instance.

```
tar –zxvf pi-apigee-4.1.tar.gz
```

    2. To verify that the tool successfully installed, run the `ls` command at the command prompt.

Result:

You see the Ping Identity directories and the build `.tgz` file.

The following table lists the available directories.

Directory	Description
`bin`	Contains the following scripts: ■ `deploy.sh` : The script to deploy the PingIntelligence policy. ■ `undeploy.sh` : The script to undeploy the PingIntelligence policy. ■ `status.sh` : Reports the deployment status and configured Apigee credentials.

Copyright © 2025 Ping Identity Corporation

Directory	Description
`certs`	Contains the certificate `ase.pem` file that is shipped by default with ASE. Make sure that the certificate applied for the ASE data port matches with the certificate present in the `ase.pem` certificate file for self-signed certificates. For more information, see Preparing to deploy the PingIntelligence shared flow.
`client_certs`	Contains the Apigee root certificate authority (CA) certificate that is copied to this location when mutual TLS (MTLS) is enabled. **Note** This feature requires ASE version 5.1.3 or later.
`lib`	`.jar` files and various dependencies. Do not edit the contents of this directory.
`policy`	Contains the shared flows: ■ `request_shared_flow_custom.zip` ■ `request_shared_flow_kvm.zip` ■ `response_shared_flow_custom.zip` ■ `response_shared_flow_kvm.zip`
`config`	Contains the `apigee.properties` file.
`logs`	Contains the log and status files.

3. To configure the PingIntelligence policy tool after installation, edit the `apigee.properties` file and set the necessary properties.

   For more information, see Apigee properties file configuration.

Apigee properties file configuration

The `apigee.properties` file is required for all sideband Apigee configurations.

The properties file is used to set properties for the PingIntelligence policy tool after installation. You can optionally configure it to capture user information. You can find the file in the `/pingidentity/apigee/config/` directory.

The following tables describe the variables in the file.

Copyright © 2025 Ping Identity Corporation

General variables

Variable	Description
`configuration_store`	**Where to store the ASE token.** **The possible values are `kvm` and `custom`. The default is `custom`.** **When you choose `custom`, the ASE token is configured inside the PingIntelligence policy and uploaded to Apigee Edge directly. When `kvm` is chosen, the ASE token is stored in the KVM store.**
`apigee_url`	**URL to connect to Apigee Edge.** **Note** If your Apigee installation is on a private cloud, change the URL to the one that matches your Apigee management server API IP:Port or hostname with protocol.
`apigee_username`	**The username to connect to Apigee Edge.**
`apigee_password`	**The password to connect to Apigee Edge.**
`apigee_environment`	**The target environment for the PingIntelligence shared flow.**
`apigee_organization`	**The target organization for the PingIntelligence shared flow.**
`ase_host_primary`	**The ASE primary host IP address and port or hostname and port.**
`ase_host_secondary`	**The ASE secondary host IP address and port or hostname and port.** **Note** This field cannot be left empty. In a testing environment, you can provide the same IP address for primary and secondary ASE host.
`ase_ssl`	**Enable or disable SSL communication between Apigee Edge and ASE. The default value is `true`.**
`ase_sideband_token`	**Configure the ASE token generated in Preparing to deploy the PingIntelligence shared flow.**

Copyright © 2025 Ping Identity Corporation

Variable	Description
`enable_mtls`	Enable or disable mutual authentication between ASE and the Apigee API gateway. The default is `false`. **Note** This feature requires ASE version 5.1.3 or later.
`mtls_password`	When mutual TLS (MTLS) is enabled, the password to access the keystore. **Note** If the private key is password protected, then the keystore and private key password must be the same.

Configuration properties to extract user information

Variable	Description
`enable_oauth_policy`	Choose whether to use the PingIntelligence OAuth policy to extract `user_info` or not. Possible values are `true` or `false`. The default value is `false`. • When set to `true`, the PingIntelligence OAuthPolicy is executed and `user_info` is sent to ASE. • When set to `false`, the PingIntelligence OAuthPolicy is not executed. The `user_info` is captured from an existing custom OAuthPolicy, if available, and sent to ASE. In both the cases, even if authorization token is deleted by the gateway, `user_info` and token information are still sent to ASE. For more information on the PingIntelligence OAuth policy, see Extract user information from access tokens.
`enable_async`	Choose synchronous or asynchronous mode between the gateway and ASE. In synchronous mode, the gateway waits for a response from ASE before forwarding the request to backend server. If asynchronous mode is enabled, the gateway forwards the request to the backend server without waiting for the response from ASE. The ASE passively logs the request and forwards it to ABS for attack analysis. It performs attack detection without blocking of attacks. Possible values are `true` or `false`.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`access_token_position`	Location of `access_token` in the API request. Possible values are `header` or `queryparam`. The default value is `header`. It is used in the OAuthPolicy. `access_token_position=queryparam` **Note** At present, Apigee supports only the `Bearer` prefix in an authorization header.
`access_token_variable`	A variable to hold `access_token` value, as shown in the following example. `access_token_variable=access_token => -H "access_token: Rft3dqrs56Blirls56a"` The default value is `Authorization`. It is used in the OAuthPolicy.
`username_key_mapping`	This is used in the PingIntelligence policy to set the key of `username` attribute in `access_token` info. The default value is `username`.
`client_id_key_mapping`	This is used in the PingIntelligence policy to set the key of `client_id` attribute in the `access_token` info. The default value is `client_id`.

Timeout configurations

Variable	Description
`connect_timeout`	Connection timeout in milliseconds between Apigee API gateway and PingIntelligence ASE.
`io_timeout`	Read timeout in milliseconds between Apigee API gateway and PingIntelligence ASE.

Copyright © 2025 Ping Identity Corporation

Variable	Description
`keepalive_timeout`	**Connection keepalive timeout between Apigee API gateway and PingIntelligence ASE. Make sure that `enable_keepalive` to `true` in `ase.conf` for the keep-alive configuration to take effect.** **Note** Make sure that the `enable_sideband_keepalive` is set to `true` in `ase.conf` file for keep-alive connection between Apigee API gateway and ASE. For more information, see Sideband ASE configuration using the `ase.conf` file.

Note

Backslashes (`\`) are not supported in `username` and `client_id` values.

The following is a sample `apigee.properties` file.

Copyright © 2025 Ping Identity Corporation

```
# Copyright 2020 Ping Identity Corporation. All Rights Reserved.
# Ping Identity reserves all rights in The program as delivered. Unauthorized use, copying,
# modification, reverse engineering, disassembling, attempt to discover any source code or
# underlying ideas or algorithms, creating other works from it, and distribution of this
# program is strictly prohibited. The program or any portion thereof may not be used or
# reproduced in any form whatsoever except as provided by a license without the written
# consent of Ping Identity.  A license under Ping Identity's rights in the Program may be
# available directly from Ping Identity.

# KVM Mode kvm/custom
configuration_store=custom
# Apigee management server URL
apigee_url=https://api.enterprise.apigee.com
# Apigee management server username
apigee_username=
# Apigee management server username
apigee_password=
# Apigee environment to which it should be deployed
apigee_environment=prod
# Apigee organization name
apigee_organization=

# ASE Primary Host  <IP/Host>:<port>
ase_host_primary=
# ASE Secondary Host  <IP/Host>:<port>
ase_host_secondary=
# ASE SSL status
ase_ssl=true
# ASE sideband authentication token
ase_sideband_token=none

# Enable OAuth Policy (allowed values: true | false)

enable_oauth_policy=false
# Enable async (allowed values: true | false)
enable_async=true

# Position of Access Token (allowed values: header | queryparam)
access_token_position=header
# access_token_position=header, access_token_variable=Authorization => -H "Authorization: Bearer
Rft3dqrs56Blirls56a"
# access_token_position=header, access_token_variable=access_token => -H "access_token:
Rft3dqrs56Blirls56a"
# access_token_position=queryparam, access_token_variable=access_token => ...?
access_token=Rft3dqrs56Blirls56a
access_token_variable=Authorization

# username key mapping in access_token. This is the key of username in access_token attributes
username_key_mapping=username
# client_id key mapping in access_token. This is the key of client_id in access_token attributes
client_id_key_mapping=client_id

# connection timeout between Apigee and ASE. Value is in milliseconds
connect_timeout=5000
```

Copyright © 2025 Ping Identity Corporation

```
# read timeout between Apigee and ASE. Value is in milliseconds
io_timeout=5000
# keepalive timeout between Apigee and ASE. Value is in milliseconds
# set enable_keepalive to true in ase.conf for the below configuration to take effect
keepalive_timeout=30000
```

> **Note**
>
> If `configuration_store` is set to `custom`, the configuration will be embedded into the PingIntelligence policy.
> If `configuration_store` is set to `kvm`, the configuration is pushed to a key-value map store while deploying the policy and is retrieved during policy execution.

Optional: Configuring MTLS security

Add optional MTLS security for the sideband connection between ASE and the Apigee API gateway.

About this task

> **Note**
>
> This feature requires ASE version 5.1.3 or later.

To configure MTLS security:

Steps

1. Copy the Apigee TLS certificates to the deployment tool `client_certs` folder:
    1. Copy all Apigee TLS certificates to the `/opt/pingidentity/apigee-policy/client_certs/client.pem` file.

        > **Note**
        >
        > If a certificate is part of a chain, then you must copy all certificates in the chain to the `/opt/pingidentity/apigee-policy/client_certs/client.pem` file. The certificates must be in order, and the last certificate must be a root certificate or an intermediate certificate signed by a root certificate.

    2. Copy the private key file (`.key`) to `/opt/pingidentity/apigee-policy/client_certs/key.pem`.
    3. Create a `myKeystore.p12` file under `opt/pingidentity/apigee-policy/client_certs/` using the `openssl` utility:

        ```
        openssl pkcs12 -export -out "myKeystore.p12" -inkey key.pem -in client.pem -name rootCert -
        passout "pass:ABC123" -passin "pass:ABC123"
        ```


Copyright © 2025 Ping Identity Corporation

Note

- If the private key is encrypted or password protected, perform one of the following:
  - Add the private key password in the `mtls_password=` option in the `apigee.properties` file.
  - Remove the password requirement by using `openssl` utility:

    ```
    bash:$. cp private.key private.key.secure bash:$. openssl rsa -in server.key.secure -out server.key
    ```
- When creating the `myKeystore.p12` file, another password can be specified. However, that password should be the same as the private key password configured in the `mtls_password=` option in the `apigee.properties` file.

2. **Copy the Apigee root certificate authority (CA) certificate to `/opt/pingidentity/ase/config/client_certs/client.pem` in ASE.**

   1. **Add the certificate to ASE:**

      ```
      bash $: cp Apigee_root_cert.pem /opt/pingidentity/ase/config/client_certs/client.pem
      ```

   2. **Restart ASE.**

Note

- `/opt/pingidentity/apigee-policy/client_certs/client.pem` contains the TLS certificate as a PEM file (either a certificate signed by a CA or a file containing a chain of certificates where the last certificate is signed by a CA).
- `/opt/pingidentity/apigee-policy/client_certs/key.pem` contains a private key as a PEM. Apigee Edge supports key sizes up to 2048 bits with an optional passphrase.
- PEM files comply with the X.509 format. If a certificate or private key is not defined by a PEM file, it can be converted to a PEM file by using utilities such as `openssl`. If the files are text files, they use one of the following formats:

  ```
  -----BEGIN CERTIFICATE-----
  ```

  ```
  -----END CERTIFICATE-----
  ```

  ```
  -----BEGIN ENCRYPTED PRIVATE KEY-----
  ```

  ```
  -----END ENCRYPTED PRIVATE KEY-----
  ```

Resetting timeout configurations

You can reset the timeout configurations after you have deployed the PingIntelligence policy.

About this task

Copyright © 2025 Ping Identity Corporation

There are two ways to reset the timeout configurations:

- Undeploy the policy and reset the values in the `apigee.properties` file and redeploy the PingIntelligence policy.

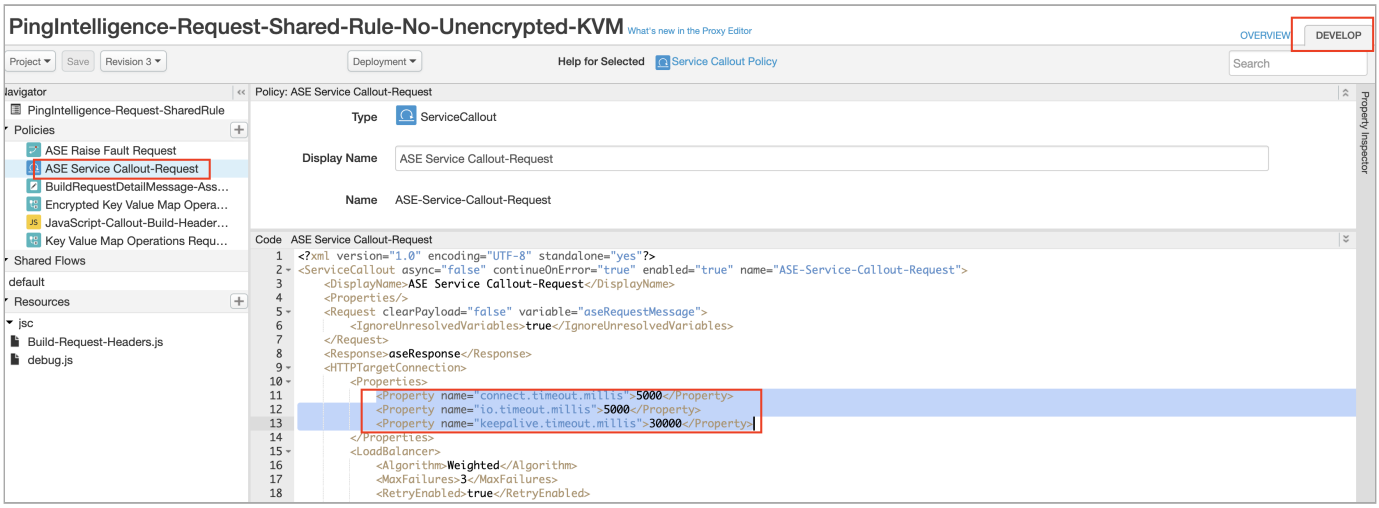
  For more information on undeploying the policy, see Changing the deployed policy mode.

- Update the values in the Apigee Edge Management UI.

To update the timeout configurations in the Apigee Edge Management UI:

Steps

1. In the Apigee Edge Management UI, go to the Shared Flows page.
2. On the Shared Flows page, open PingIntelligence-Request-SharedRule.
3. In the left navigation pane, under Policies, click ASE Service Callout-Request.
4. Click the Develop tab as shown in the screenshot.



5. Change the timeout values under HTTPTargetConnection and save the changes.
6. Repeat steps 4 and 5 for PingIntelligence-Response-Shared-Rule.

Extract user information from access tokens

PingIntelligence for APIs provides the `OAuthPolicy.xml` policy to capture user information from the requests sent to Apigee gateway.

The policy verifies the access token from the bundled Apigee OAuth server and extracts details like username and client ID and other request metadata. It can verify access tokens provided as part of a request header or a query parameter.

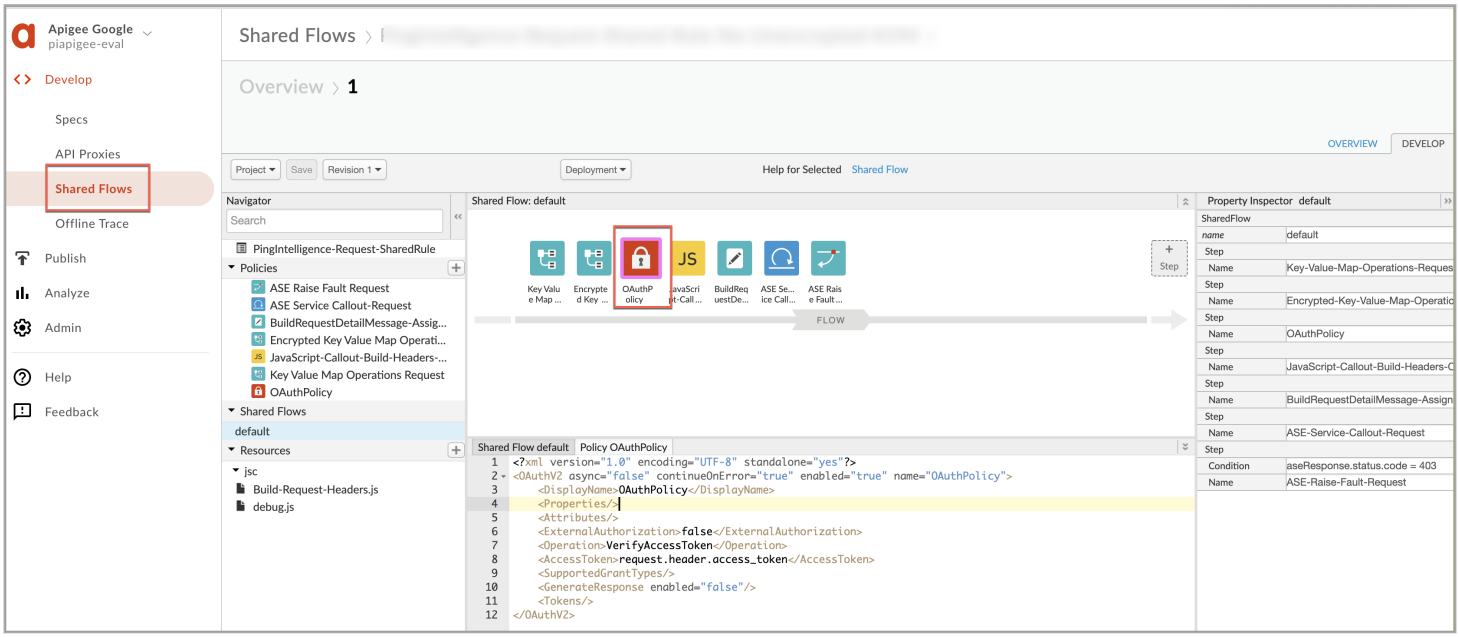
The OAuthPolicy extracts request metadata tagged to an access token. The policy should be executed before the PingIntelligence policy that builds the ASE request message, which captures the username and client ID from the metadata extracted by OAuthPolicy.

The OAuthPolicy can be attached using a flow hook or a flow callout. For more information, see Deploying the PingIntelligence policy for flow hook and Deploying the PingIntelligence policy for flow callout.

Copyright © 2025 Ping Identity Corporation

You should deploy `OAuthPolicy.xml` using a Flow CallOut policy to leverage the flexibility of applying on a per API basis. For more information, see Configuring PingIntelligence flow callout in Apigee.

The following screen capture illustrates the PingIntelligence shared flow with OAuthPolicy.



> **Note**
>
> At present, the OAuthPolicy supports extraction of user information from access tokens generated by Apigee bundled OAuth server only.

Configure `apigee.properties` file to capture the user information

Additionally set the configuration properties in `apigee.properties` file to extract the user information using the PingIntelligence OAuthPolicy. For more information, see Configure apigee.properties file to extract user information.

> **Note**
>
> If a custom OAuth policy is used in place of PingIntelligence OAuthPolicy, then configure the `enable_oauth_policy` variable in `apigee.properties` to `false`.

Deploying the PingIntelligence policy

Use the PingIntelligence automated policy tool to deploy the shared flow.

Using the PingIntelligence automated policy tool, you deploy the shared flow either by the flow hook or the flow callout policy, which is configured in the command line. Choose either the included ASE self-signed certificate or a certificate authority (CA)-signed certificate.

Copyright © 2025 Ping Identity Corporation

Flow hook

Deploying the PingIntelligence policy for flow hook
About this task

With a flow hook, the PingIntelligence shared flow is applied to all APIs in the environment of an organization.

Steps

1. Deploy the shared flow:

   Choose from:

   - Deploy with self-signed certificate: To deploy the PingIntelligence policy with self-signed certificate, run the following command.

```
/opt/pingidentity/pi/apigee/bin/deploy.sh -fh
Checking Apigee connectivity
Apigee connectivity ... success
Generating policies

Deploying PI Apigee policy Flow Hook

1) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
2) PingIntelligence-Config-KVM status ... not-applicable
3) ASE Server status ... deployed
4) Truststore status ... deployed
5) Upload pem file status ... deployed
6) Cache status ... deployed
7) Request policy upload status ... deployed
8) Response policy upload status ... deployed
9) Request policy deployment status ... deployed
10) Response policy deployment status ... deployed
11) Preproxy Flow hook status ... deployed
12) Postproxy Flow hook status ... deployed

Deployment of PI Policy finished successfully
```


Copyright © 2025 Ping Identity Corporation

- Deploy with CA-signed certificate: To deploy the PingIntelligence policy with CA-signed certificate, run the following command.

```
/opt/pingidentity/pi/apigee/bin/deploy.sh -fh -ca

Checking Apigee connectivity
Apigee connectivity ... success
Generating policies

Deploying PI Apigee policy Flow Hook

1) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
2) PingIntelligence-Config-KVM status ... not-applicable
3) ASE Server status ... deployed
4) Truststore status ... not-applicable - running using CA signed certificate
5) Upload pem file status ... not-applicable - running using CA signed certificate
6) Cache status ... deployed
7) Request policy upload status ... deployed
8) Response policy upload status ... deployed
9) Request policy deployment status ... deployed
10) Response policy deployment status ... deployed
11) Preproxy Flow hook status ... deployed
12) Postproxy Flow hook status ... deployed

Deployment of PI Policy finished successfully
```

2. After you deploy the flow hook using the PingIntelligence tool, to verify the status of the deployment, run the following command:

```
/opt/pingidentity/pi/apigee/bin/status.sh
Checking Apigee connectivity
Apigee connectivity ... success

Checking the PI Apigee Policy Flow Hook deployment status

1) PingIntelligence-Config-KVM status ... not applicable
2) PingIntelligence-Encrypted-Config-KVM status ... not applicable
3) ASE target status ... deployed
4) Cache status ... deployed
5) Truststore status ... deployed
6) Request Policy status ... deployed
7) Response Policy status ... deployed
8) Preproxy hook status ... deployed
9) Postproxy hook status ... deployed

PI Apigee Policy is already installed
```

Copyright © 2025 Ping Identity Corporation

Flow callout

Deploying the PingIntelligence policy for flow callout
About this task

In the flow callout, the PingIntelligence policy is applied on a per-API basis in the environment of an organization.

Steps

1. Deploy the shared flow:

   Choose from:

   - Deploy with self-signed certificate: To deploy the PingIntelligence policy with self-signed certificate, run the following command.

```
/opt/pingidentity/pi/apigee/bin/deploy.sh -fc
Checking Apigee connectivity
Apigee connectivity ... success
Generating policies

Deploying PI Apigee policy Flow Call Out

1) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
2) PingIntelligence-Config-KVM status ... not-applicable
3) ASE Server status ... deployed
4) Truststore status ... deployed
5) Upload pem file status ... deployed
6) Cache status ... deployed
7) Request policy upload status ... deployed
8) Response policy upload status ... deployed
9) Request policy deployment status ... deployed
10) Response policy deployment status ... deployed
11) Preproxy Flow call out status ... deployed
12) Postproxy Flow call out status ... deployed

Deployment of PI Policy finished successfully
```


Copyright © 2025 Ping Identity Corporation

- Deploy with CA-signed certificate: To deploy the PingIntelligence policy with CA-signed certificate, run the following command.

```
bin/deploy.sh -fc -ca

Checking Apigee connectivity
Apigee connectivity ... success
Generating policies

Deploying PI Apigee policy Flow Call Out

1) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
2) PingIntelligence-Config-KVM status ... not-applicable
3) ASE Server status ... deployed
4) Truststore status ... not-applicable - running using CA signed certificate
5) Upload pem file status ... not-applicable - running using CA signed certificate
6) Cache status ... deployed
7) Request policy upload status ... deployed
8) Response policy upload status ... deployed
9) Request policy deployment status ... deployed
10) Response policy deployment status ... deployed
11) Preproxy Flow call out status ... deployed
12) Postproxy Flow call out status ... deployed

Deployment of PI Policy finished successfully
```

2. After deploying the flow callout using the PingIntelligence tool, to verify the status of the deployment, run the following command:

```
/opt/pingidentity/pi/apigee/bin/status.sh
Checking Apigee connectivity
Apigee connectivity ... success

Checking the PI Apigee Policy Flow Call Out deployment status

1) PingIntelligence-Config-KVM status ... not applicable
2) PingIntelligence-Encrypted-Config-KVM status ... not applicable
3) ASE target status ... deployed
4) Cache status ... deployed
5) Truststore status ... deployed
6) Request Policy status ... deployed
7) Response Policy status ... deployed
8) Preproxy call out status ... deployed
9) Postproxy call out status ... deployed

PI Apigee Policy is already installed
```

Copyright © 2025 Ping Identity Corporation

Configuring PingIntelligence flow callout in Apigee

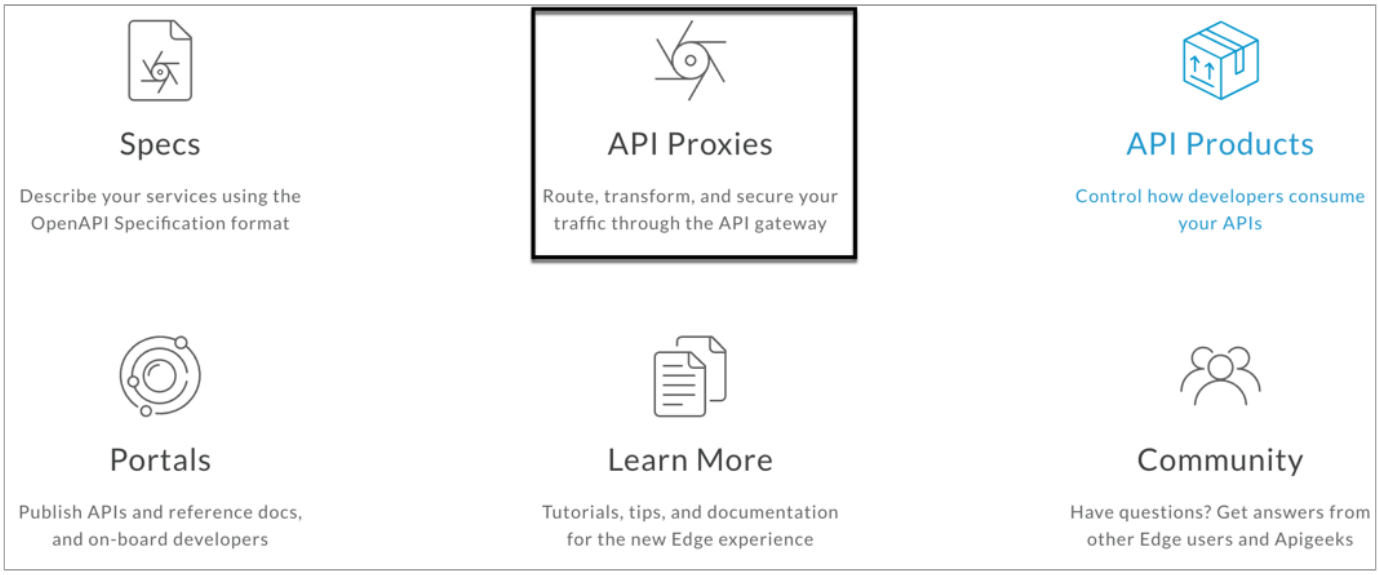
After deploying the Flow Callout policy using PingIntelligence, configure the PingIntelligence for APIs shared flow.

About this task

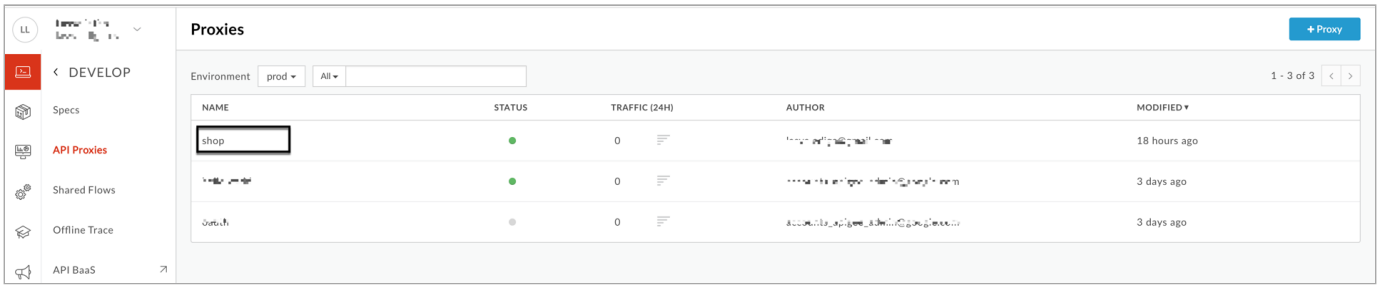
> **Note**
>
> The following steps are the same for Flow Callout for both request and response.

Steps

1. **Sign on to your Apigee Edge account and select API Proxies.**



2. **Click the API name for which you want to apply the policy.**

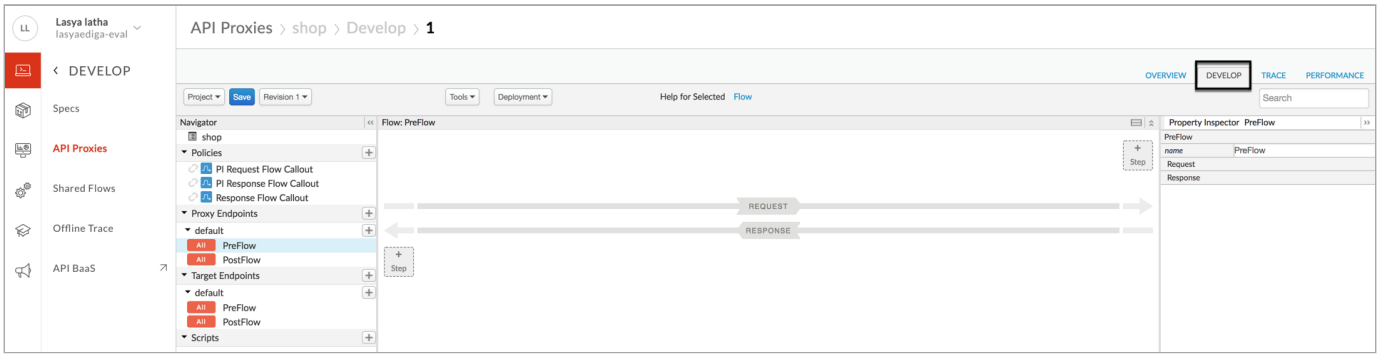


Result:

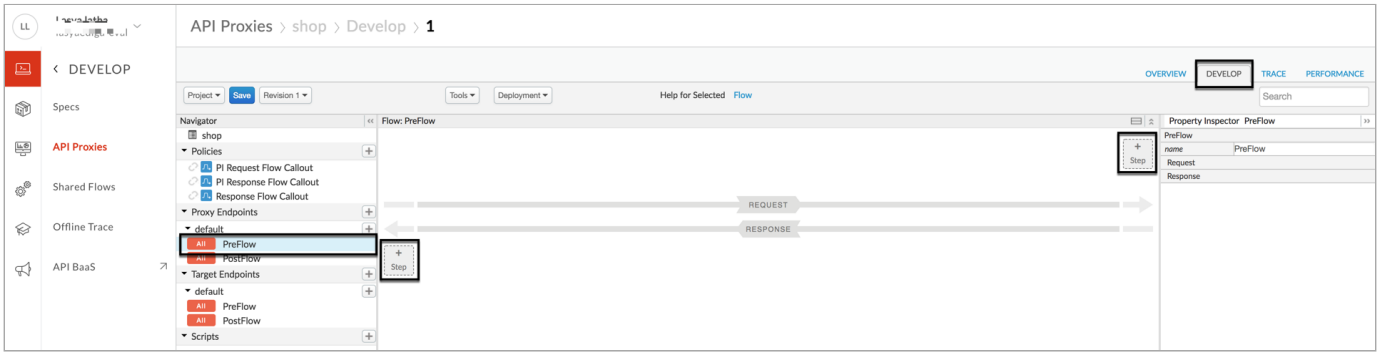
The Develop page opens.

3. **On the Develop page, click the Develop tab.**

Copyright © 2025 Ping Identity Corporation



4. On the Develop tab under Proxy Endpoints, select PreFlow and click + Step for request.



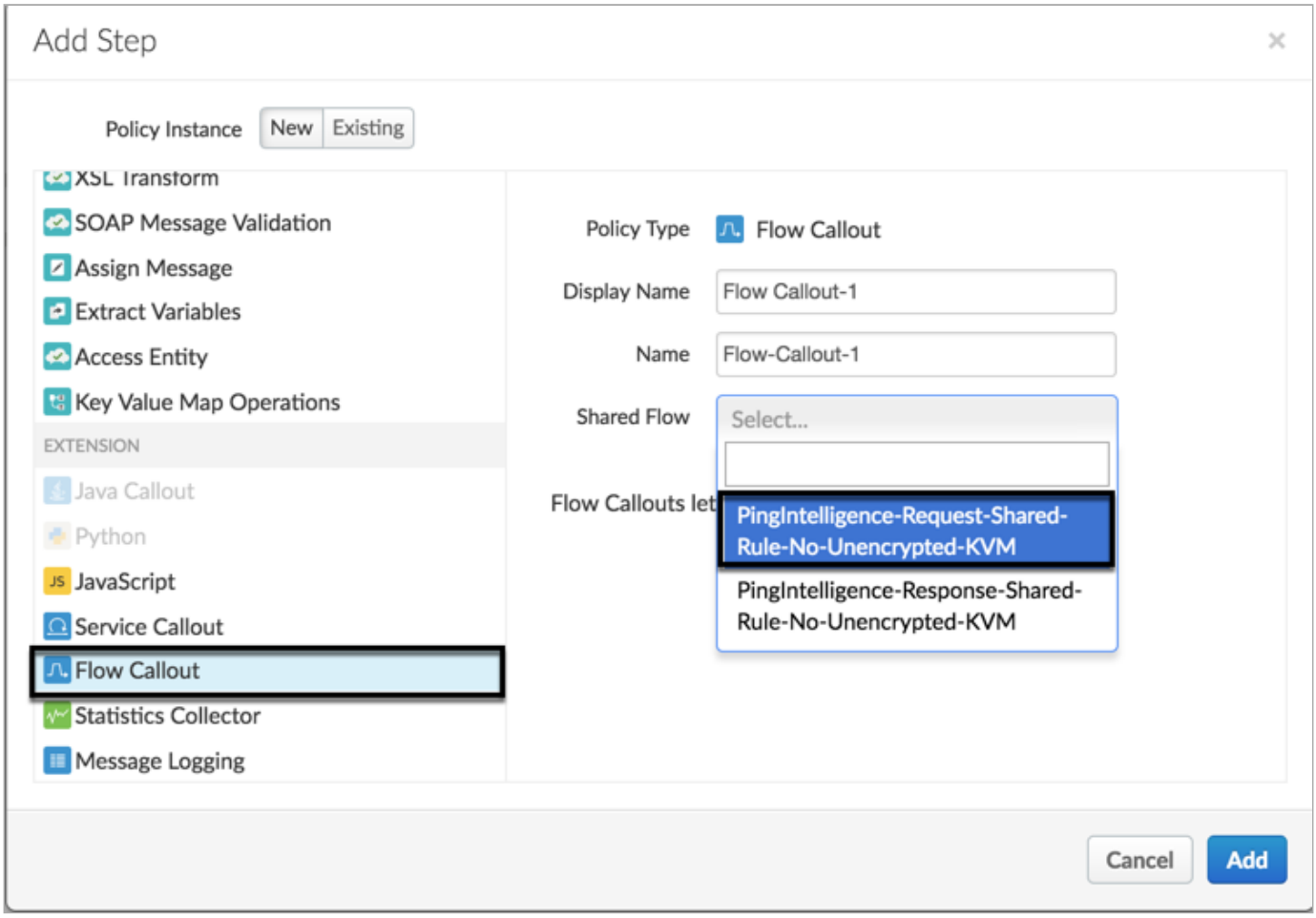
Result:

The Add Step page opens.

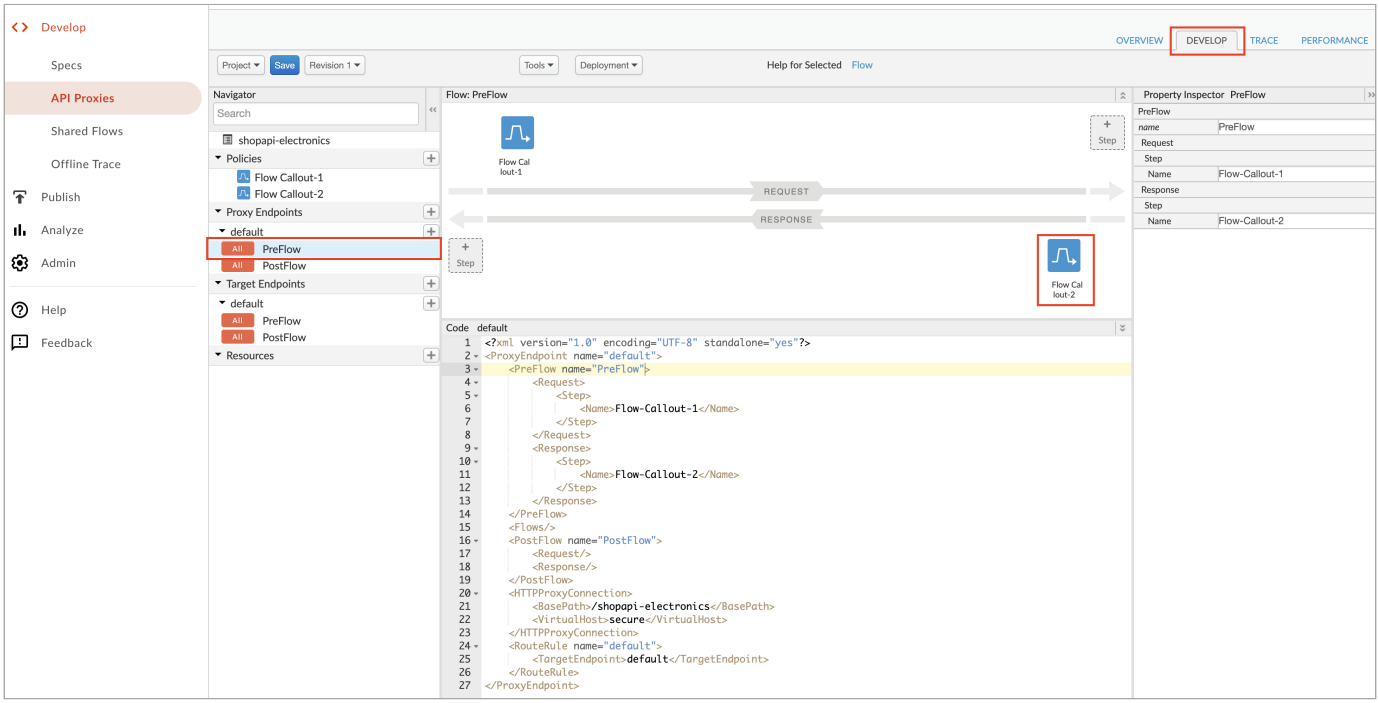
5. On the Add Step page, select Flow Callout.

6. In the Shared Flow list, select the Request rule and click Add.

Copyright © 2025 Ping Identity Corporation



7. On the Develop tab under Proxy Endpoints, select PreFlow and click+ Step for response.

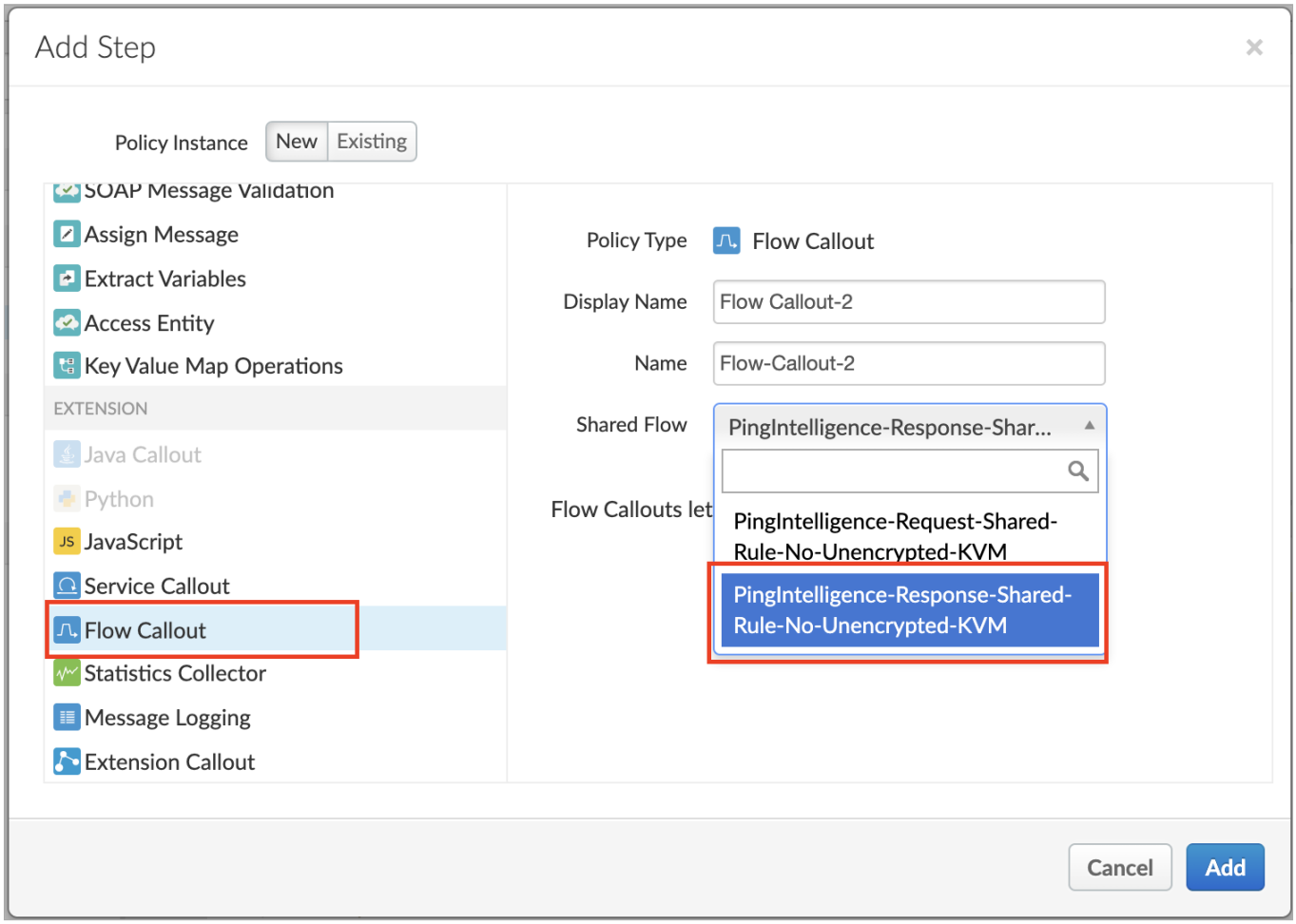


Result:

Copyright © 2025 Ping Identity Corporation

The Add Step page opens.

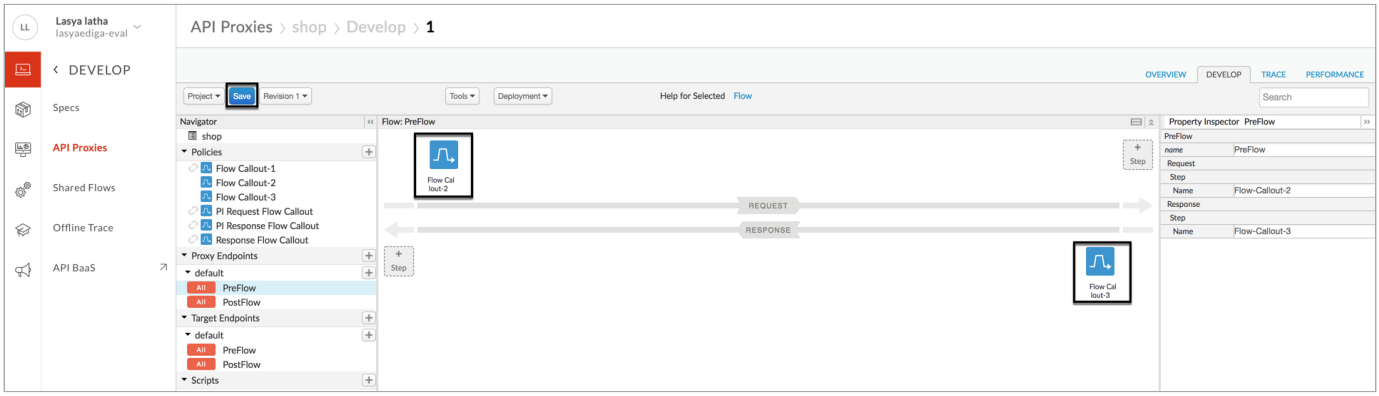
8. On the Add Step page, click Flow Callout.
9. In the Shared Flow list, select the Response rule and click Add.



Result:

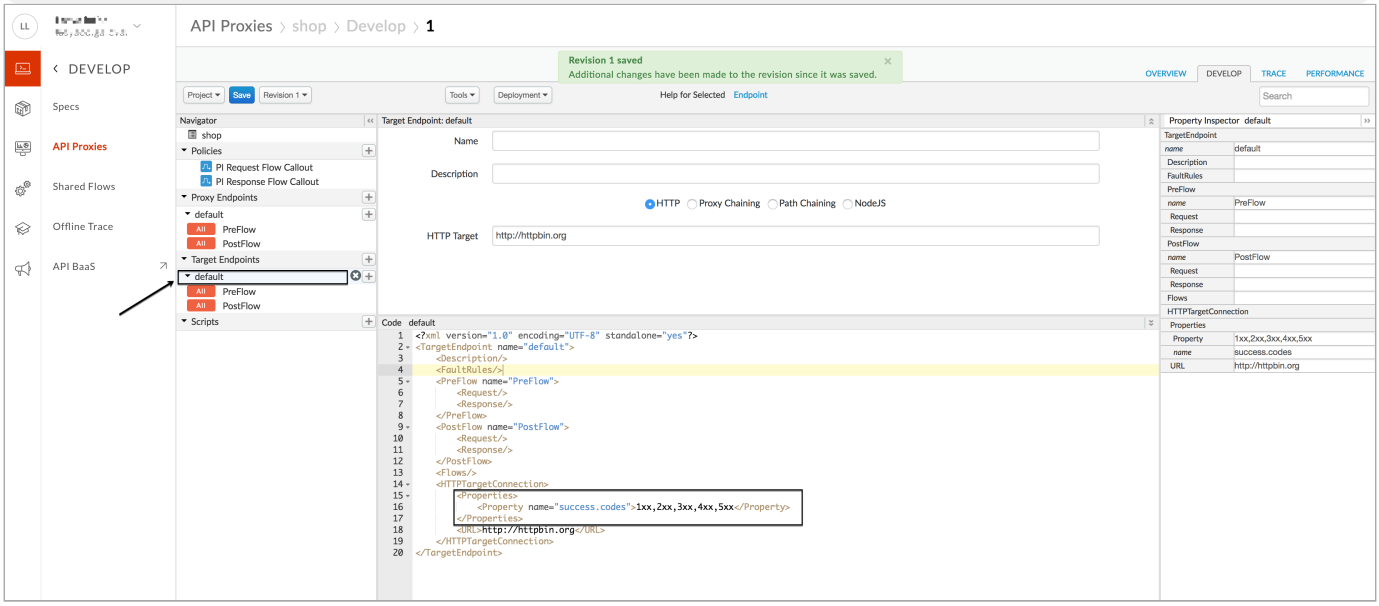
The request and response rules are added.

10. Click Save.



11. Click default and enter the following lines in the `<HTTPTargetConnection>` tag.

```
<Properties>
            <Property name="success.codes">1xx,2xx,3xx,4xx,5xx</Property>
</Properties>
```



12. To save the revision, click Save.

Changing the deployed policy mode

You can change the type of policy deployed from flow hook to flow callout, or flow callout to flow hook using the PingIntelligence policy tool.

About this task

> **Tip**
>
> Using the following steps, you can also change the use of security certificate from self-signed to certificate authority (CA)-signed, or from CA-signed to self-signed.

To change the type of policy:

Copyright © 2025 Ping Identity Corporation

Steps

1. To undeploy the deployed policy, run one of the following commands based on the policy and certificate used:

   Choose from:

   - Undeploy a Flow Hook policy using a self-signed certificate.

```
/opt/pingidentity/pi/apigee/bin/undeploy.sh -fh
Checking Apigee connectivity
Apigee connectivity ... success

Undeploying PI Apigee policy Flow Hook

1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... undeployed
12) Cache status ... undeployed

Undeployment of PI Policy finished successfully
```

   - Undeploy a Flow Hook policy using a CA-signed certificate.

```
opt/pingidentity/pi/apigee/bin/deploy.sh -fh -ca

Checking Apigee connectivity
Apigee connectivity ... success

Undeploying PI Apigee policy Flow Hook

1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... not-applicable - running using CA signed certificate
12) Cache status ... undeployed

Undeployment of PI Policy finished successfully
```

Copyright © 2025 Ping Identity Corporation

- Undeploy a Flow Callout policy using a self-signed certificate.

```
/opt/pingidentity/pi/apigee/bin/undeploy.sh -fc
Checking Apigee connectivity
Apigee connectivity ... success

Undeploying PI Apigee policy Flow Call Out

1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... undeployed
12) Cache status ... undeployed

Undeployment of PI Policy finished successfully
```

- Undeploy a Flow Callout policy using a CA-signed certificate.

```
opt/pingidentity/pi/apigee/bin/deploy.sh -fc -ca

Checking Apigee connectivity
Apigee connectivity ... success

Undeploying PI Apigee policy Flow Call Out

1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... not-applicable - running using CA signed certificate
12) Cache status ... undeployed

Undeployment of PI Policy finished successfully
```

Next steps

To deploy the other policy, see Flow Hook or Flow Call Out.

Copyright © 2025 Ping Identity Corporation

Add API definitions to ASE

Add the API definition files to help API Security Enforcer (ASE) extract metadata from API traffic, set decoys to trap intruding attacks, perform health checks on backend servers, and so on.

The API definitions also help the artificial intelligence (AI) engine to build AI models to detect any Indicators of Attacks (IoAs) on APIs. After the policy has been deployed to Apigee using the PingIntelligence automated policy tool, add APIs to ASE.

For more information on defining APIs, see:

- API naming guidelines
- Defining an API using API JSON configuration file in sideband mode

For more information on ASE sideband deployment, see Sideband ASE.

Undeploying the PingIntelligence policy

Using the PingIntelligence automated policy tool, you can undeploy the shared flow either by flow hook or the flow callout policy.

To undeploy the PingIntelligence policy:

Flow hook

Undeploying the PingIntelligence policy for Flow Hook
Steps

- Undeploy the PingIntelligence policy:

  Choose from:

  - Undeploy with self-signed certificate: Run the following command to undeploy the PingIntelligence policy with self-signed certificate.

    ```
    /opt/pingidentity/pi/apigee/bin/undeploy.sh -fh
    ```

  - Undeploy with certificate authority (CA)-signed certificate: Run the following command to undeploy the PingIntelligence policy with a CA-signed certificate.

    ```
    /opt/pingidentity/pi/apigee/bin/undeploy.sh -fh -ca
    ```

Copyright © 2025 Ping Identity Corporation

Flow callout

Undeploying the PingIntelligence policy for Flow Callout
Steps

- Undeploy the PingIntelligence policy:

  Choose from:

  - Undeploy with self-signed certificate: Run the following command to undeploy the PingIntelligence policy with self-signed certificate.

    ```
    /opt/pingidentity/pi/apigee/bin/undeploy.sh -fc
    ```

  - Undeploy with certificate authority (CA)-signed certificate: Run the following command to undeploy the PingIntelligence policy with CA-signed certificate.

    ```
    /opt/pingidentity/pi/apigee/bin/undeploy.sh -fc -ca
    ```

Troubleshooting mismatch of self-signed certificates

If the ASE certificate is changed after the deployment of the PingIntelligence policy and it doesn't match with the certificate present in the `ase.pem` certificate file, you might encounter Secure Sockets Layer (SSL)-related issues.

About this task

To resolve these issues:

Steps

1. Undeploy the PingIntelligence policy by following either of the two options as applicable:

   Choose from:

   - Undeploy PingIntelligence policy for Flow Hook with self-signed certificate
   - Undeploy PingIntelligence policy for Flow Call Out with self-signed certificate

2. To obtain the correct certificate to match what's in the `ase.pem` file, run the following command.

   ```
   # openssl s_client -showcerts -connect  <ASE IP address>:<port no>  </dev/null 2>/dev/null | openssl x509 -outform PEM > ase.pem
   ```

3. Paste the correct certificate in the `/opt/pingidentity/pi/apigee/certs/ase.pem` file.

4. Redeploy the PingIntelligence policy by following either of the two options as applicable:

Copyright © 2025 Ping Identity Corporation

Choose from:

- Deploy PingIntelligence policy for Flow Hook with self-signed certificate
- Deploy PingIntelligence policy for Flow Call Out with self-signed certificate

> **Note**
>
> Make sure that the `ase_ssl` parameter in `/pingidentity/pi/apigee/config/apigee.properties` file is set to `true`.

AWS API Gateway integration

This integration guide discusses the deployment of PingIntelligence for APIs in a sideband configuration with an AWS API Gateway through CloudFront.

PingIntelligence for APIs provides a sideband policy that can be installed in CloudFront. The policy uses AWS Lambda functions to pass API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.

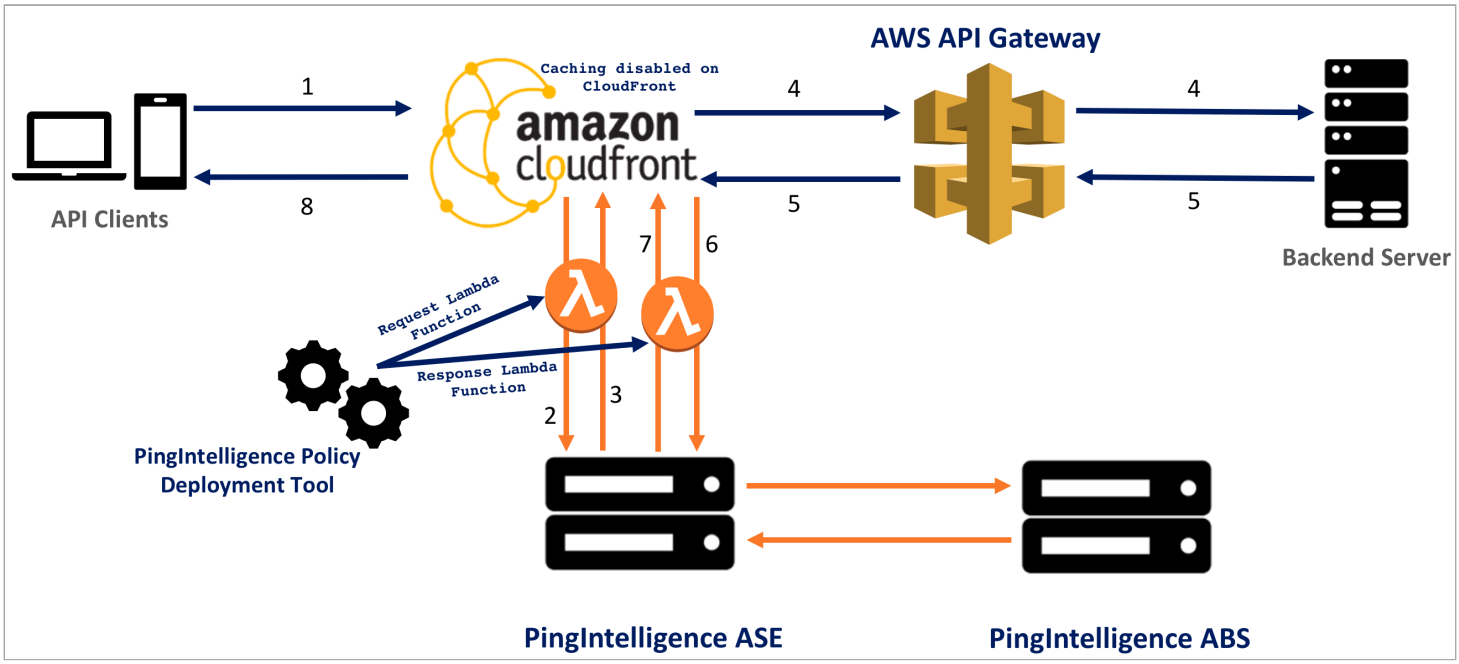
PingIntelligence for APIs provides an automated tool to deploy a the policy, which is implemented using the AWS Lambda functions. The policy requires AWS CloudFront to be present with all caching disabled. AWS Lambda functions must be initially deployed in the US-East-1 region, and the policy definition is pushed to any region with your API Gateways after the PingIntelligence policy is added.

The PingIntelligence sideband policy requires a CloudFront instance, which can be an existing or new instance.

> **Important**
>
> The default AWS Lambda memory is sufficient for up to 1000 QPS. For a larger QPS, contact Ping Identity support. See the aws.properties file for default origin response value.

The following diagram shows the logical setup of PingIntelligence API Security Enforcer (ASE) and CloudFront.

Copyright © 2025 Ping Identity Corporation



The traffic flow through the CloudFront and PingIntelligence for APIs components is as follows:

1. An incoming API client request destined for the API Gateway arrives at CloudFront.
2. A PingIntelligence AWS Lambda policy makes an API call to send the request metadata to PingIntelligence ASE.
3. ASE checks the request against a registered set of APIs and looks for the origin IP, cookie, OAuth2 token, or API key in the API Behavioral Security (ABS) artificial intelligence (AI) engine-generated deny list. If all checks pass, ASE returns a `200-OK` response to AWS Lambda. If the checks don't pass, ASE sends a `403` response code to AWS Lambda. The request information is also logged by ASE and sent to the ABS AI Engine for processing.
4. If CloudFront receives a `200-OK` response from ASE, it forwards the client request to the backend server. Otherwise, the CloudFront blocks the client when blocking is enabled for the API.
5. CloudFront receives the response from the backend server.
6. The Lambda response function makes a second API call to pass the response information to ASE.
7. ASE receives the response information and sends a `200-OK` to AWS Lambda. The response information is also logged by ASE and sent to the ABS AI Engine for processing.
8. CloudFront sends the response received from the backend server to the client.

Preparing to run the AWS policy tool

Before running the PingIntelligence AWS policy tool, complete the following prerequisites.

About this task

Before running the PingIntelligence AWS policy tool:

Steps

1. Install OpenJDK 11 on the system running the PingIntelligence policy tool.

Copyright © 2025 Ping Identity Corporation

2. Install and configure the PingIntelligence software. Refer to the PingIntelligence deployment guide for your environment.

   To deploy the PingIntelligence sideband policy, you must have an AWS admin account.

   Make sure that AWS cross-account is not used to deploy PingIntelligence policy.

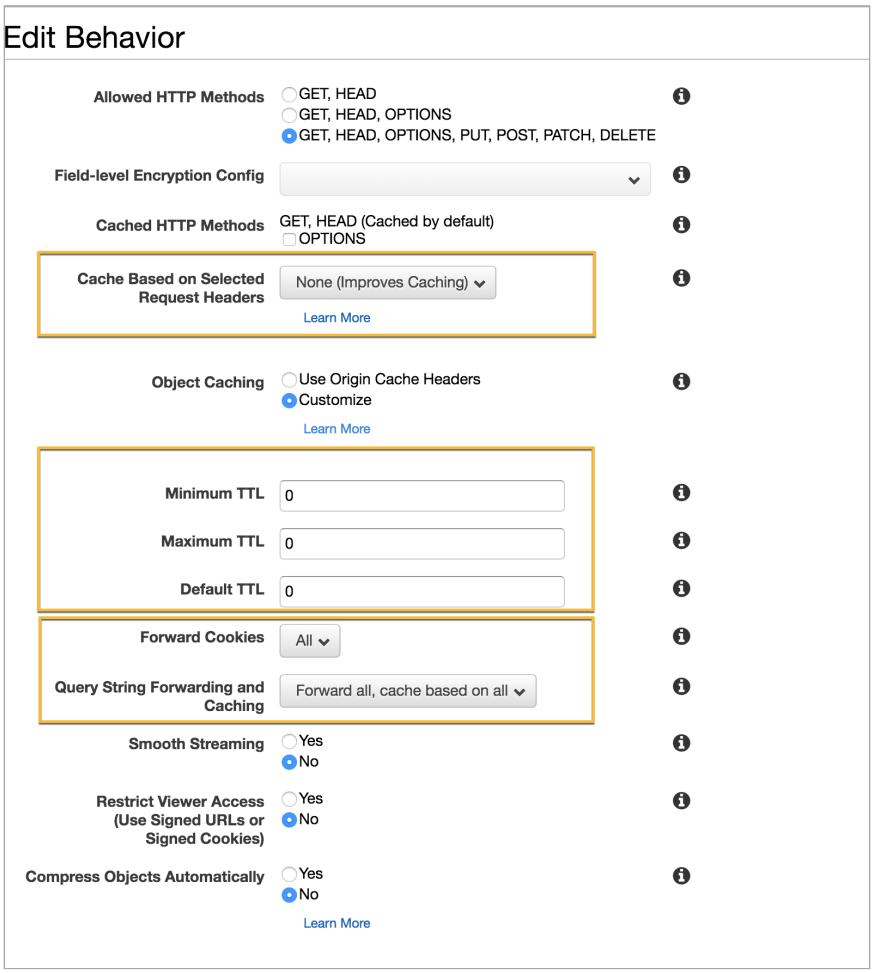
3. To update the CloudFront configuration, verify the following options are configured correctly:

   1. The PingIntelligence policy deployment tool requires that CloudFront be available with caching disabled for all CloudFront behaviors. Select None (Improves Caching) from the Cache Based on Selected Request Headers drop-down list.

   2. Confirm that Minimum TTL, Maximum TTL, and the Default TTL are set to 0.

   3. For Forward Cookies, select All from the drop-down list.

   4. Under Query String Forwarding and Caching, select Forward all, cache based on all from the drop-down list.



4. The PingIntelligence policy tool requires viewer request and origin response Lambda functions. Make sure that there is no viewer request or origin response Lambda function defined in the caching behavior.

5. Verify that ASE is in sideband mode by running the following command in the ASE command line:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh status
```

Result:

```
API Security Enforcer
status                  : started
 mode : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : enabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

Troubleshooting:

If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

6. For a secure communication between CloudFront and ASE, enable sideband authentication by entering the following command in the ASE command line:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

7. A token is required for CloudFront to authenticate with ASE. This token is generated in ASE and configured in the `aws.properties` file of the PingIntelligence automated policy tool. To generate the token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

8. Optional: For improved performance, set the `enable_sideband_keepalive` parameter to `true` in the `ase.conf` file.

   For more information, see Sideband ASE configuration using the `ase.conf` file.

Installing the PingIntelligence AWS policy tool

Download and install the PingIntelligence Amazon Web Services (AWS) policy tool.

About this task

To download and install the PingIntelligence policy tool:

Steps

1. Download the PingIntelligence policy tool to the `/opt` directory.

Copyright © 2025 Ping Identity Corporation

2. Complete the following steps to untar the policy tool:

    1. At the command prompt, enter the following command to untar the policy tool file:

    ```
    tar –zxvf  <filename>
    ```

    Example:

    ```
    tar –zxvf pi-aws-4.0.tar.gz
    ```

    2. To verify that the tool successfully installed, enter the `ls` command at the command prompt. This should list the `pingidentity` directory and the build `.tgz` file.

    The following table lists the directories.

Directory	Description
`bin`	Contains the following scripts: ■ `deploy.sh` : The script to deploy the PingIntelligence policy ■ `undeploy.sh` : The script to undeploy the PingIntelligence policy ■ `status.sh` : Reports the deployment status of the identity and access management (IAM) role and Lambda function
`lib`	`jar` files and various dependencies. Do not edit the contents of this directory.
`policy`	Contains the request and response Lambda functions: ■ `request_lambda.zip` ■ `response_lambda.zip`
`config`	Contains the `aws.properties` file
`logs`	Contains the log and status files

Configuring the AWS automated policy tool

Configure the automated Amazon Web Services (AWS) policy tool.

About this task

To configure the automated policy tool:

Steps

1. Configure the `aws.properties` file available in the `/pingidentity/pi/aws/config/` directory.

Copyright © 2025 Ping Identity Corporation

The following table describes the variables in the `aws.properties` file.

Variable	Description
`mode`	Choose the authentication mode between `keys` and `role`. **Note** If you are running the PingIntelligence policy tool from your local machine, use the `keys` mode. If you are running the tool from an EC2 instance, use the `role` mode.
`access_key`	AWS access key. This is applicable when the mode is set to `keys`.
`secret_key`	AWS secret key. This is applicable when the mode is set to `keys`.
`aws_lambda_memory`	AWS Origin Response Lambda memory in MB. Default value is 1024 MB. The memory can be configured in multiple of 64. Minimum and maximum value are 128 and 3008 respectively. For more information, see AWS Lambda Pricing.
`cloudfront_distribution_id`	The CloudFront distribution ID.
`ase_host_primary`	The API Security Enforcer (ASE) primary host Internet Protocol (IP) address and port or hostname and port.
`ase_host_secondary`	The ASE secondary host IP address and port or hostname and port. ASE secondary host receives traffic only when the primary ASE host is unreachable. **Note** This field cannot be left blank. In a testing environment, enter the same IP address for primary and secondary ASE host. If both the ASE hosts are unreachable, the request is directly sent to the backend API server.
`ase_ssl`	Enable or disable Secure Sockets Layer (SSL) communication between Lambda functions and ASE. The default value is `true`.
`ase_sideband_token`	Enter the ASE token generated during the Preparing to run the AWS policy tool, step 7.

Example:

Copyright © 2025 Ping Identity Corporation

The following is a sample `aws.properties` file:

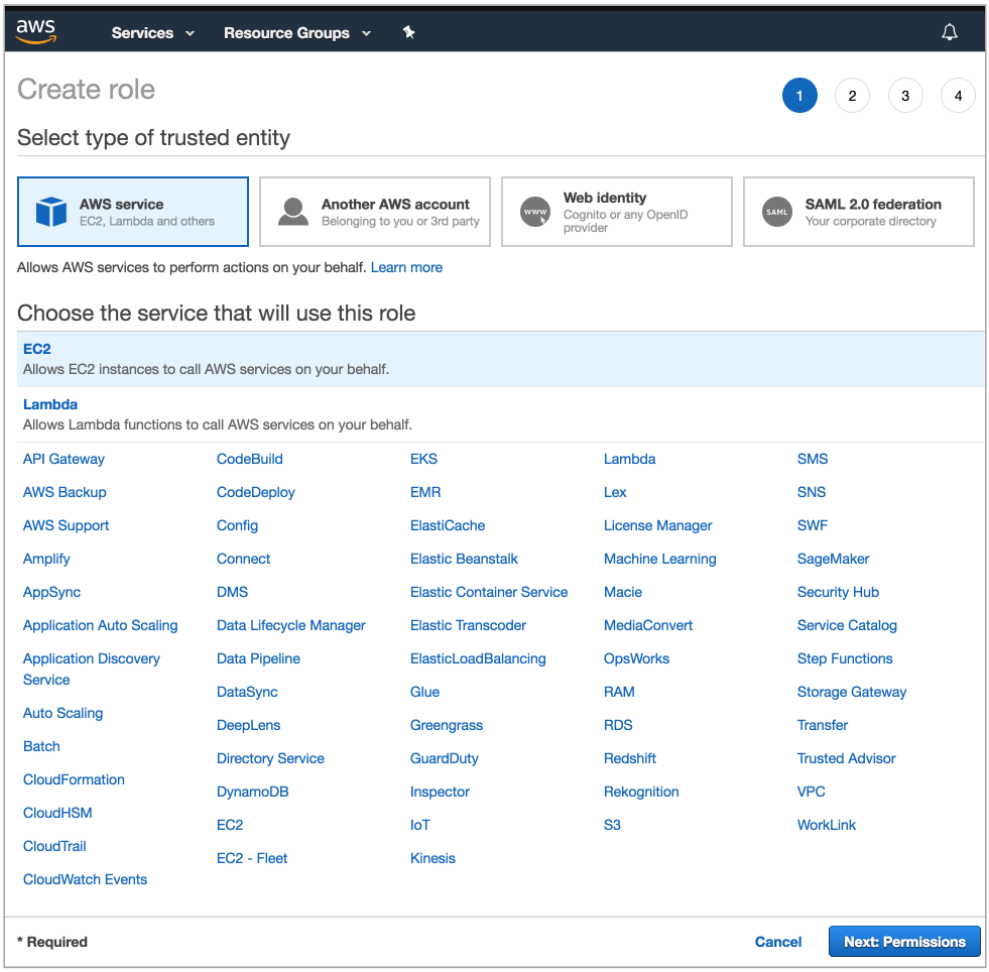
```
# Copyright 2019 Ping Identity Corporation. All Rights Reserved.
# Ping Identity reserves all rights in The program as delivered. Unauthorized use, copying,
# modification, reverse engineering, disassembling, attempt to discover any source code or
# underlying ideas or algorithms, creating other works from it, and distribution of this
# program is strictly prohibited. The program or any portion thereof may not be used or
# reproduced in any form whatsoever except as provided by a license without the written
# consent of Ping Identity.  A license under Ping Identity's rights in the Program may be
# available directly from Ping Identity.

#Authentication mode access-key & secret-key / role based access. Values can be keys or role.
mode=keys
#AWS access key
access_key=AKIAID7MDWSCUUVHMTNA
#AWS secret key
secret_key=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
#AWS Lambda memory in MB. It should be a multiple of 64. Minimum and maximum value are 128 and 3008
respectively.
aws_lambda_memory=1024
#Cloudfront distribution ID
cloudfront_distribution_id=EGQ9OEG3ZDABP

#ASE Primary Host <IP/Host>:<port>
ase_host_primary=test.elasticbeam.com
#ASE Secondary Host <IP/Host>:<port>
ase_host_secondary=test.elasticbeam.com
#ASE SSL status
ase_ssl=true
#ASE sideband authentication token
ase_sideband_token=283ded57cd5f48e6bcd8fa3ba9d2888d
```

2. If you have set the authentication `mode` as `role` in the `aws.properties` file, create a role for the EC2 instance.

   This role is required for the PingIntelligence policy deployment tool.

   1. Select EC2 as service and click on Next: Permissions button.

Copyright © 2025 Ping Identity Corporation

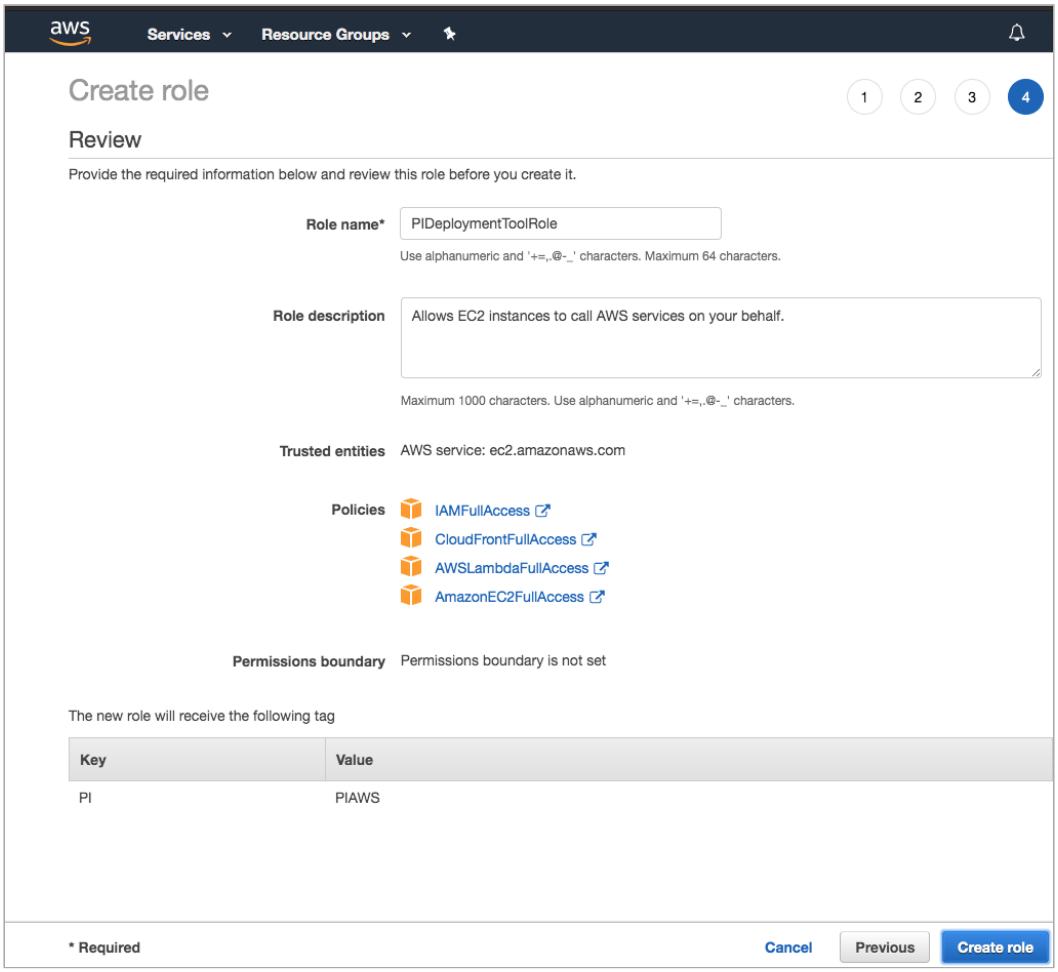


2. Choose the following four policies and provide a name for each role (for example, PIDeploymentToolRole):

   - `IAMFullAccess`
   - `AWSLambdaFullAccess`
   - `CloudFrontFullAccess`
   - `AmazonEC2FullAccess`

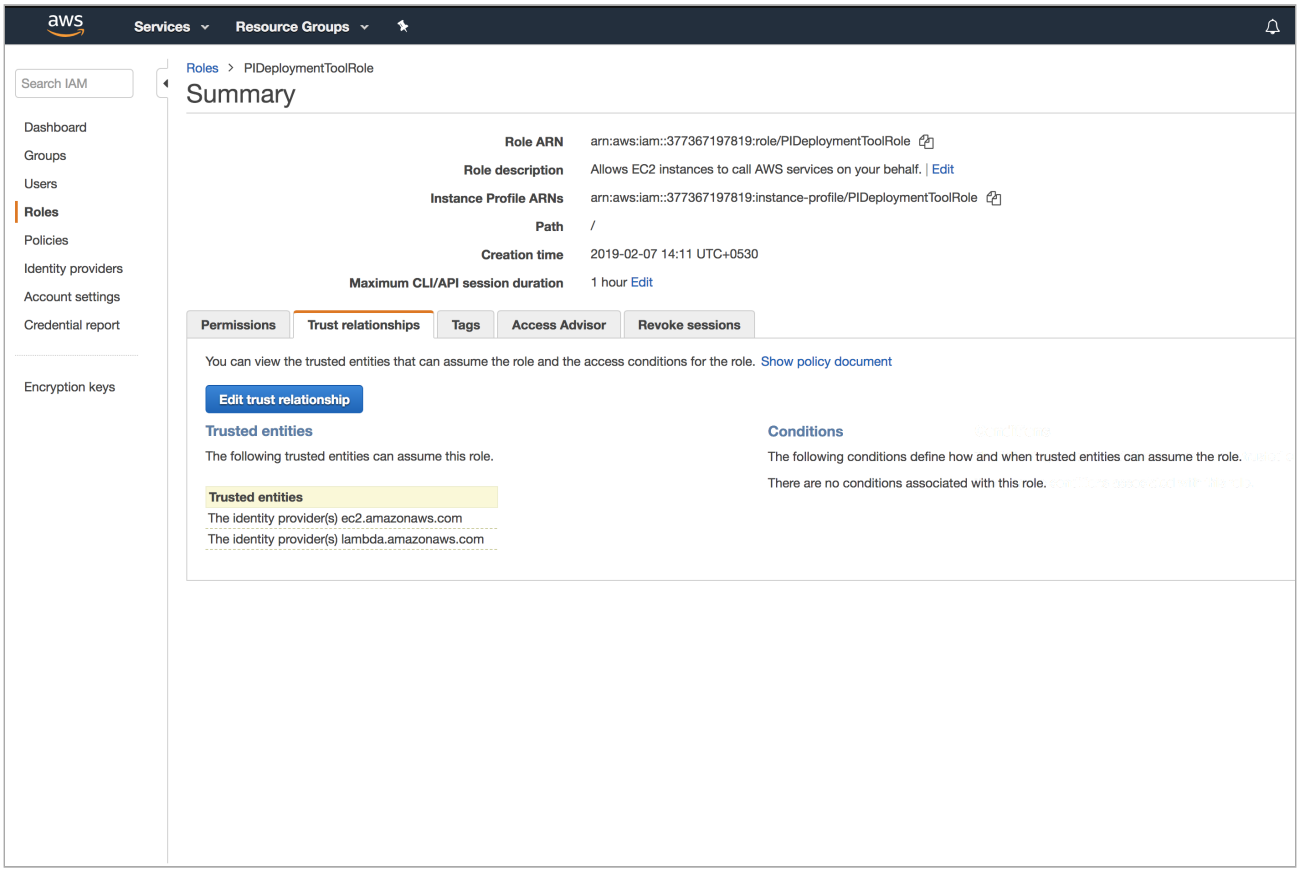
3. After providing the name, click on Create role.

Copyright © 2025 Ping Identity Corporation



4. **In the Summary page of the role that you created in step 2c, click the Trust relationships tab and then click the Edit trust relationship button.**

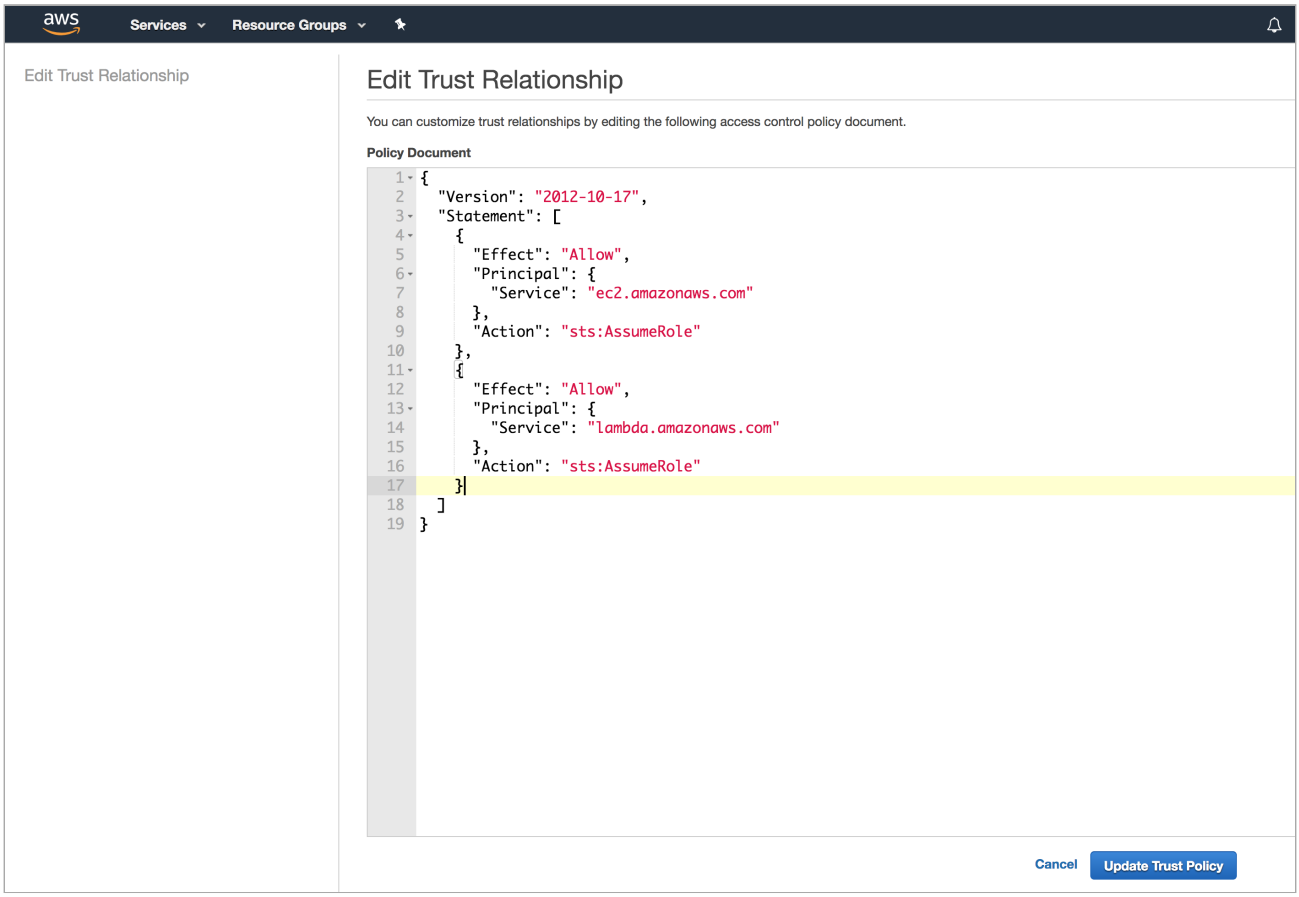
Copyright © 2025 Ping Identity Corporation



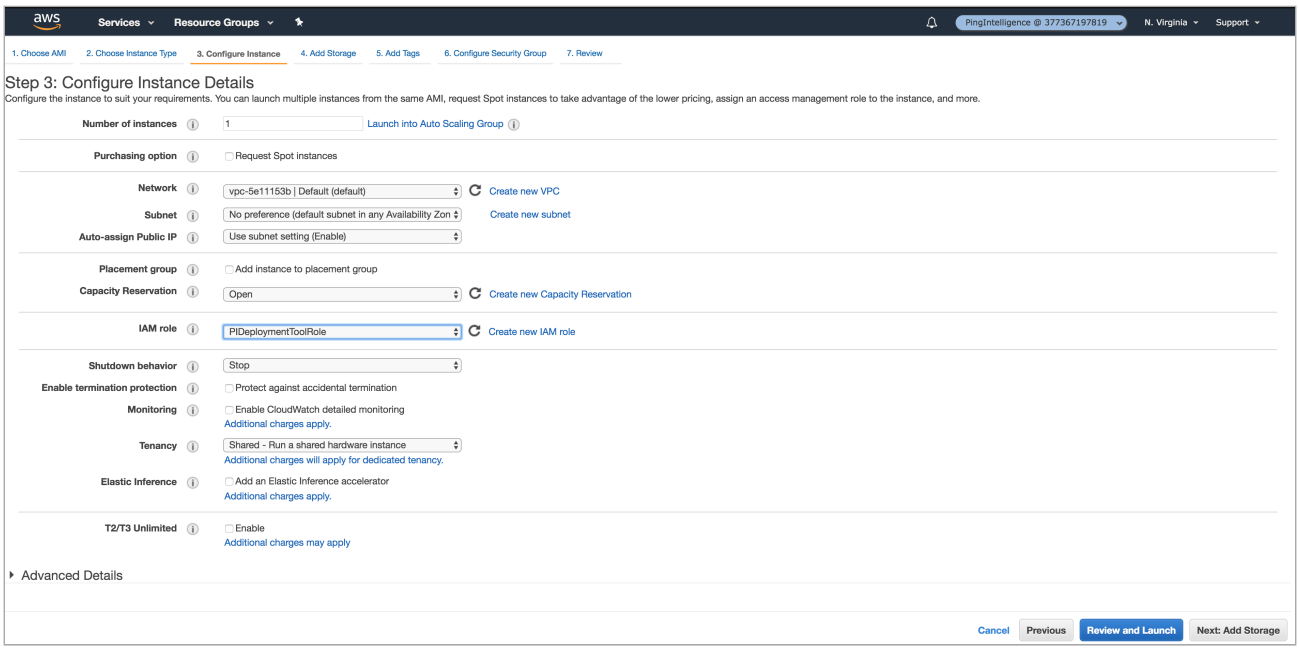
5. On the Edit Trust Relationship page, enter the following lines and click Update Trust Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```


Copyright © 2025 Ping Identity Corporation



6. **Configure the IAM role as the role that you created in step 2c (for example, PIDeploytmentToolRole).**



Copyright © 2025 Ping Identity Corporation

Deploying the PingIntelligence policy for AWS

Using the PingIntelligence Amazon Web Services (AWS) policy tool, deploy the PingIntelligence policy in AWS @Lambda in the North Virginia (US-East-1) region.

About this task

The Lambda function pushes the PingIntelligence policy to the Amazon CloudFront in the local AWS instances. ThePingIntelligence Lamba policy communicates with PingIntelligence ASE to pass request and response metadata and check whether the client request should be blocked or passed to the AWS gateway.

> **Note**
>
> At present, the policy must be initially deployed in North Virginia (US-East-1) region.

To deploy the PingIntelligence policy:

Steps

1. Run the following command:

```
/opt/pingidentity/pi/aws/bin$ deploy.sh -ca
```

> **Note**
>
> When the **`deploy.sh`** script is run without `ca` option, the policy is deployed using the self-signed certificate, which is included in the PingIntelligence policy. By the running the policy tool, the following two policies are deployed:
>
> - Request Lambda
> - Response Lambda

Result:

```
Deploying PI AWS Policy with CA-signed certificate

1) Create IAM Role named PI-Role - status... done
2) Create a policy named LambdaEdgeExecution-PI - status... done
3) Attach LambdaEdgeExecution-PI Policy to Role PI-Role... done
4) Generating policy... done
5) Deploying PI-ASE-Request Lambda... done
6) Fetching PI-ASE-Request Lambda version... done
7) Deploying PI-ASE-Response Lambda... done
8) Fetching PI-ASE-Response Lamda version... done
9) Deploying PI-ASE-Request Lamda CloudFront... done
10) Deploying PI-ASE-Response Lambda CloudFront... done

Successfully deployed PI AWS Policy.
```

2. To check the status of the PingIntelligence policy deployment, run the `status.sh` command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/pi/aws/bin$ status.sh
Checking the PI AWS Policy deployment status

1) IAM Role named PI-Role deployment - status... deployed
2) IAM Policy named LambdaEdge-PI deployment - status... deployed
3) PI-ASE-Request Lamda deployment - status... deployed
4) PI-ASE-Response Lamda deployment - status... deployed
5) PI-ASE-Request Lamda CloudFront deployment - status... deployed
6) PI-ASE-Response Lamda CloudFront deployment - status... deployed

PI AWS Policy is already installed.
```

Next steps

PingIntelligence API discovery is a process to discover, and report APIs from your API environment. The discovered APIs are reported in the PingIntelligence Dashboard. APIs are discovered when a global API JavaScript Object Notation (JSON) is defined in the ASE. For more information, see API discovery and configuration. You can edit the discovered API's JSON definition in the Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see Discovered APIs.

Integrating into your API environment

The next step is to integrate the Amazon Web Services (AWS) policy tool into your API environment.

About this task

After the policy deployment is complete, refer to the following topics for next steps.

Also refer to the ASE and ABS Admin Guides.

Steps

- Refer to:
  - ASE port information
  - API naming guidelines
  - Adding APIs to ASE in Defining an API using API JSON configuration file in sideband mode.

  You can add individual APIs or you can configure a global API.
  - Connect ASE and ABS
- After adding APIs to PingIntelligence, train the API model.

  The training of an API model is executed in the ABS AI engine. The following topics provide a high-level overview of the process:
  - Train your API model.
  - Generate and view the REST API reports using Postman.
  - View PingIntelligence for APIs Dashboard.

Copyright © 2025 Ping Identity Corporation

Uninstalling the CloudFront sideband policy

Remove the PingIntelligence Amazon Web Services (AWS) policy with the undeploy tool, which detaches the policy from CloudFront.

About this task

To undeploy the policy:

Steps

1. Run the following command:

   ```
   /opt/pingidentity/pi/aws/bin$ undeploy.sh
   ```

   Result:

   ```
   Undeploying PI AWS Policy

   1) Fetching PI-ASE-Request Lambda version... done
   2) Fetching PI-ASE-Response Lamda version... done
   3) Undeploy PI-ASE-Request Lamda CloudFront... done
   4) Undeploy PI-ASE-Response Lamda CloudFront... done
   5) Undeploy PI-ASE-Request Lamda... done
   6) Undeploy PI-ASE-Response Lamda... done
   7) Detaching IAM Role named PI-Role from policy LambdaEdgeExecution-PI - status... done
   8) Deleting  IAM Role named PI-Role - status... done
   9) Deleting policy named LambdaEdgeExecution-PI - status... done

   Successfully undeployed PI AWS Policy.
   ```

   Troubleshooting:

   The time required to detach the policy from CloudFront varies depending on the CloudFront region where the policy is deployed. It is common to encounter intermediate error messages like the following during the course of uninstallation:

   ```
   Lambda was unable to delete lambda:us-east-1:377367197819:function:PI-ASE-Request-E2PLLTN1FCYDB3:5
   because it is a replicated function.
   Please see our documentation for Deleting Lambda@Edge Functions and Replicas.
   ```

   If such error occurs then re-execute the `undeploy.sh` after a gap of 30 minutes.

2. To search and verify the deletion of PingIntelligence Lambda functions, use the `cloud_front_id` from the aws.properties file

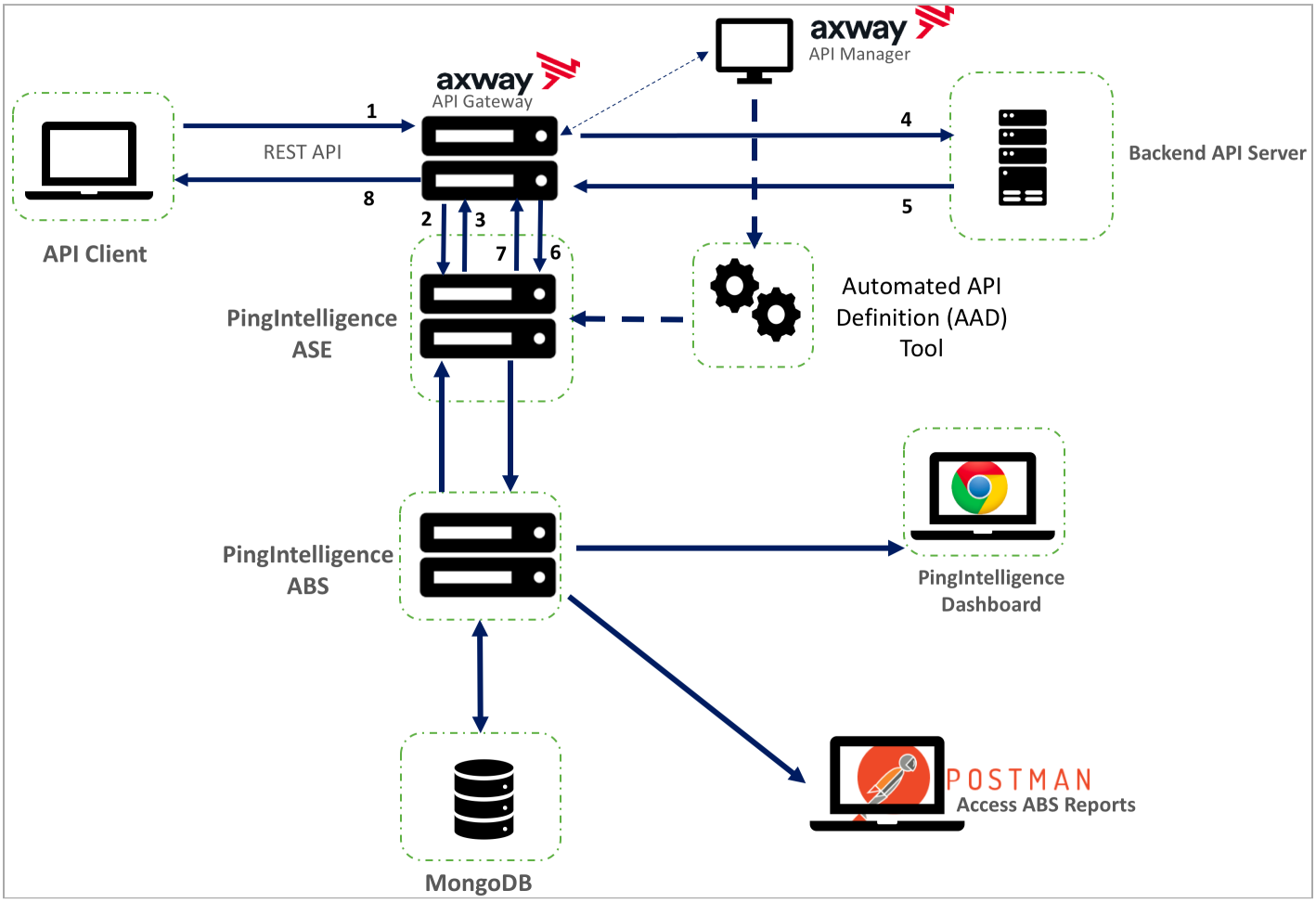
Axway sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with an Axway API Gateway.

Copyright © 2025 Ping Identity Corporation

A PingIntelligence policy is installed in the Axway API Gateway and passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking. PingIntelligence 4.0 software adds support for reporting and attack detection based on usernames captured from token attributes.

The following diagram shows the complete deployment:



The following is the traffic flow through Axway and PingIntelligence for APIs components.

1. Client sends an incoming request to Axway.
2. Axway makes an API call to send the request metadata to API Security Enforcer (ASE).
3. ASE checks the request against a registered set of APIs and checks the origin IP, cookie, API Key, or OAuth2 token in the PingIntelligence artificial intelligence (AI) engine-generated deny list. If all checks pass, ASE returns a `200-OK` response to the Axway. If not, a different response code is sent to Axway. The request information is also logged by ASE and sent to the AI engine for processing.
4. If Axway receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, the Gateway optionally blocks the client.
5. The response from the backend server is received by Axway.
6. Axway makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.

Copyright © 2025 Ping Identity Corporation

7. ASE receives the response information and sends a `200-OK` to Axway.
8. Axway sends the response received from the backend server to the client.

Preparing to configure the Axway API Gateway

Complete the following before configuring the Axway API Gateway.

About this task

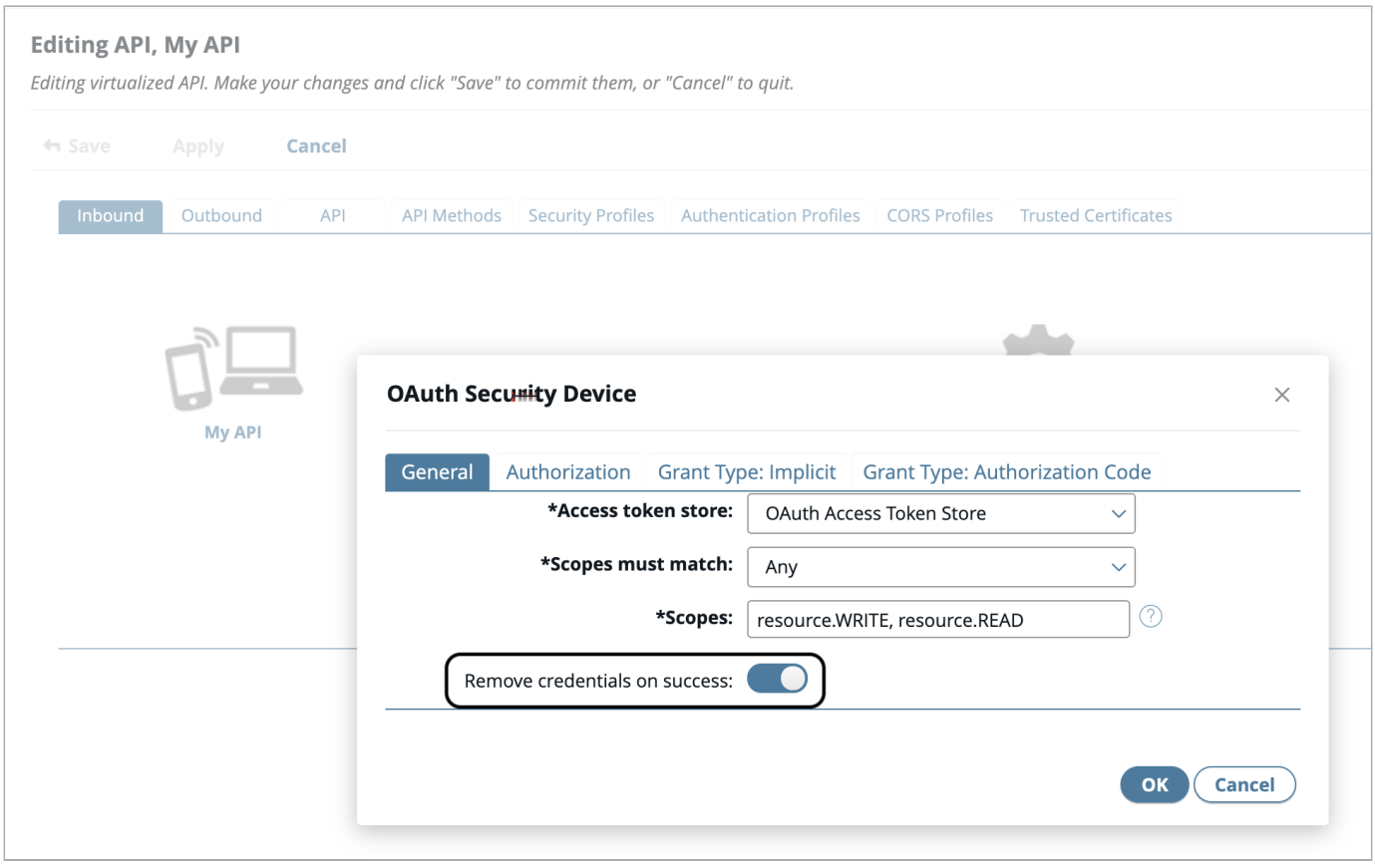
To connect PingIntelligence API Security Enforcer (ASE) with the Axway API Gateway:

Steps

1. Confirm the Axway version is 7.5.3 or higher.

   PingIntelligence works with Axway 7.5.3 or higher.

2. Optional: To detect username-based attacks, make sure that the OAuth token store is configured in Axway.



3. Install and configure the PingIntelligence software.

   Refer to the PingIntelligence deployment guide for your environment type.

4. Verify that ASE is in sideband mode by running the following ASE command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh status
```

Result:

```
API Security Enforcer
status                  : started
 mode : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : enabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

Troubleshooting:

If ASE is not in sideband mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

5. For a secure communication between Axway and ASE, enable sideband authentication by entering the following ASE command:

   ```
   # ./bin/cli.sh enable_sideband_authentication -u admin –p
   ```

6. Generate sideband authentication token by entering the following ASE command. Save the generated authentication token for further use.

   A token is required for Axway to authenticate with ASE.

   ```
   # ./bin/cli.sh -u admin -p admin create_sideband_token
   ```

7. If you are using AAD to automate API definition updates on PingIntelligence, open the following ports:
   - The management port to fetch API definitions from Axway. The default port is 8075.
   - Port 8010 in ASE for AAD to add API definitions.
8. Import the Axway policy in Axway Policy Studio.
9. Deploy the Axway policy.
10. Import the APIs from the Management virtual machine (VM) to Axway API Manager.

Configuring Axway Policy Studio

To deploy the PingIntelligence policy, you must first configure Axway Policy Studio.

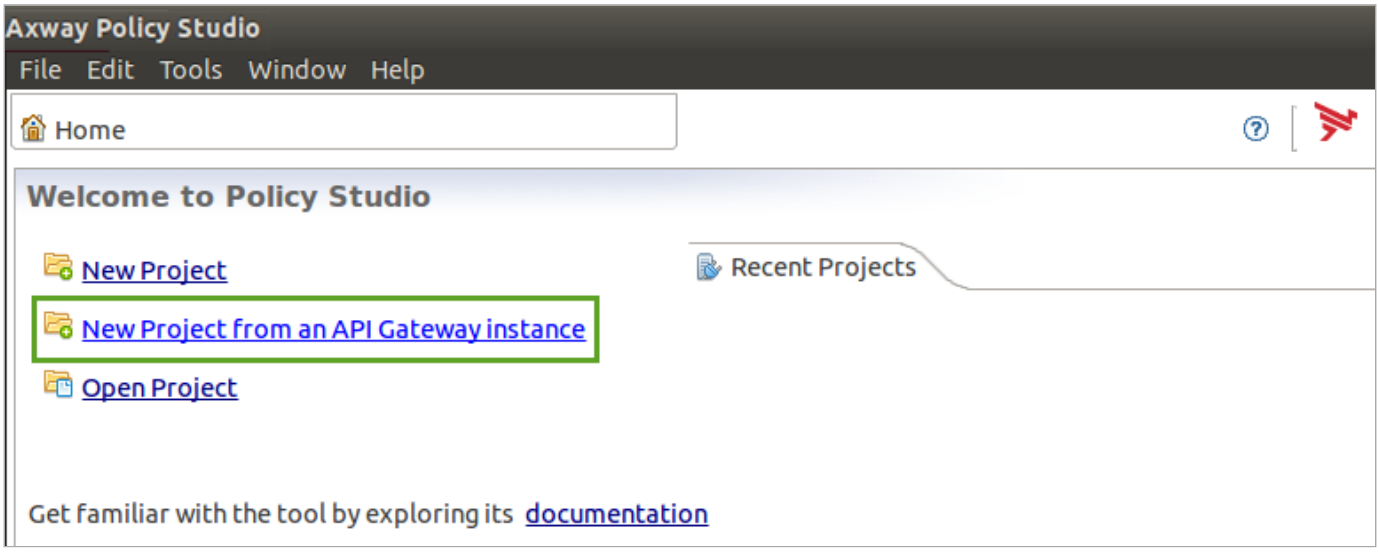
About this task

Copyright © 2025 Ping Identity Corporation

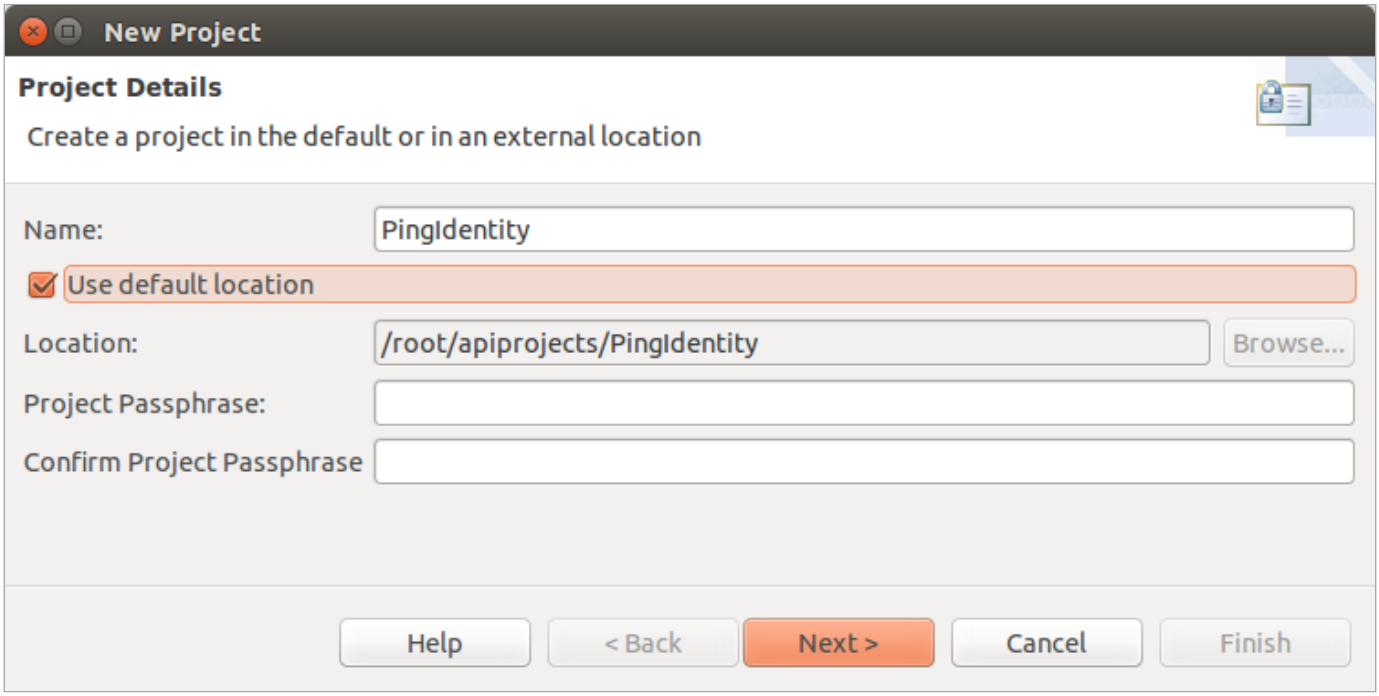
To configure Axway Policy Studio:

Steps

1. **Launch Axway Policy Studio and click New Project from an API Gateway instance.**

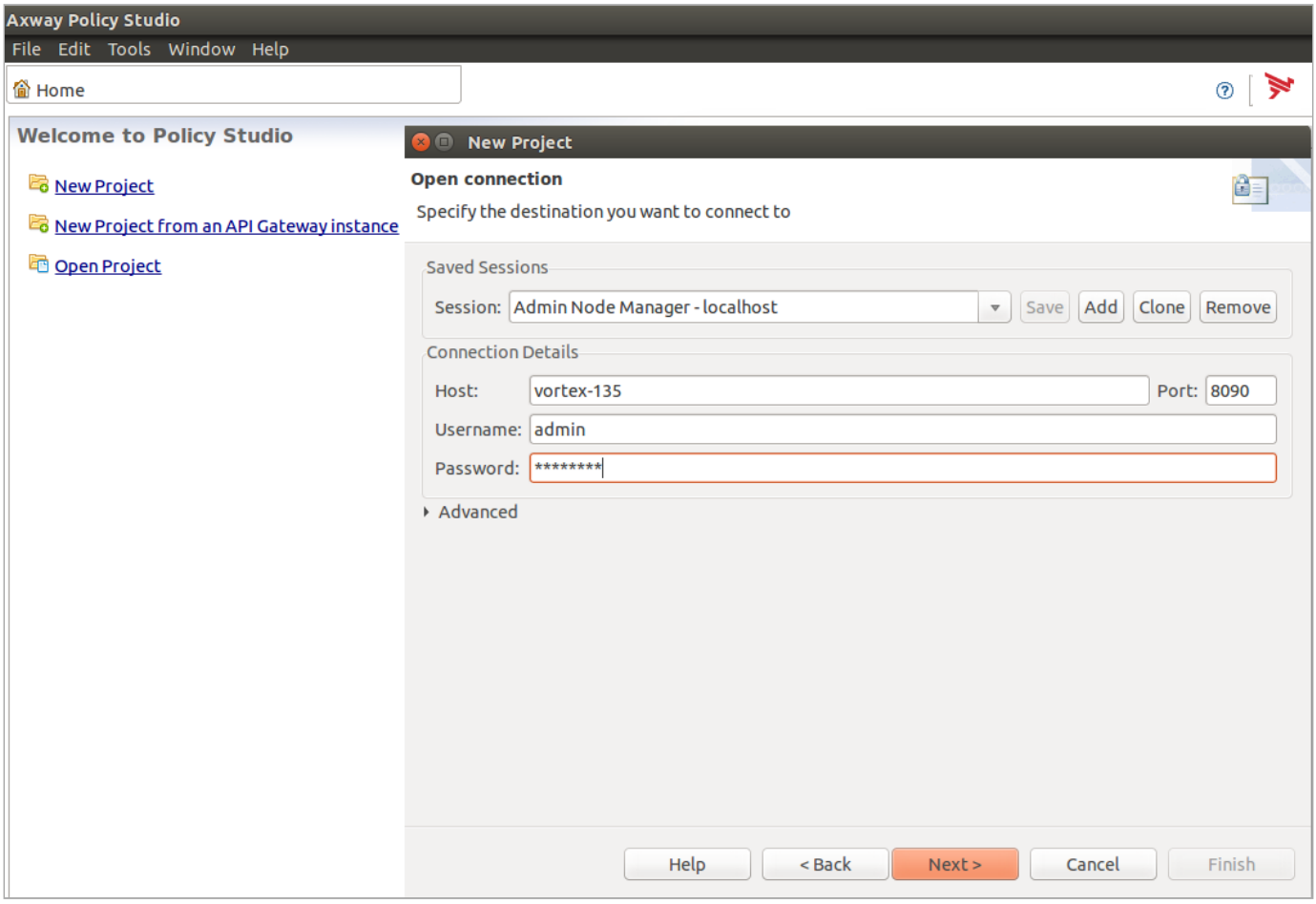


2. **In the New Project pop-up window, enter the details and click Next >.**

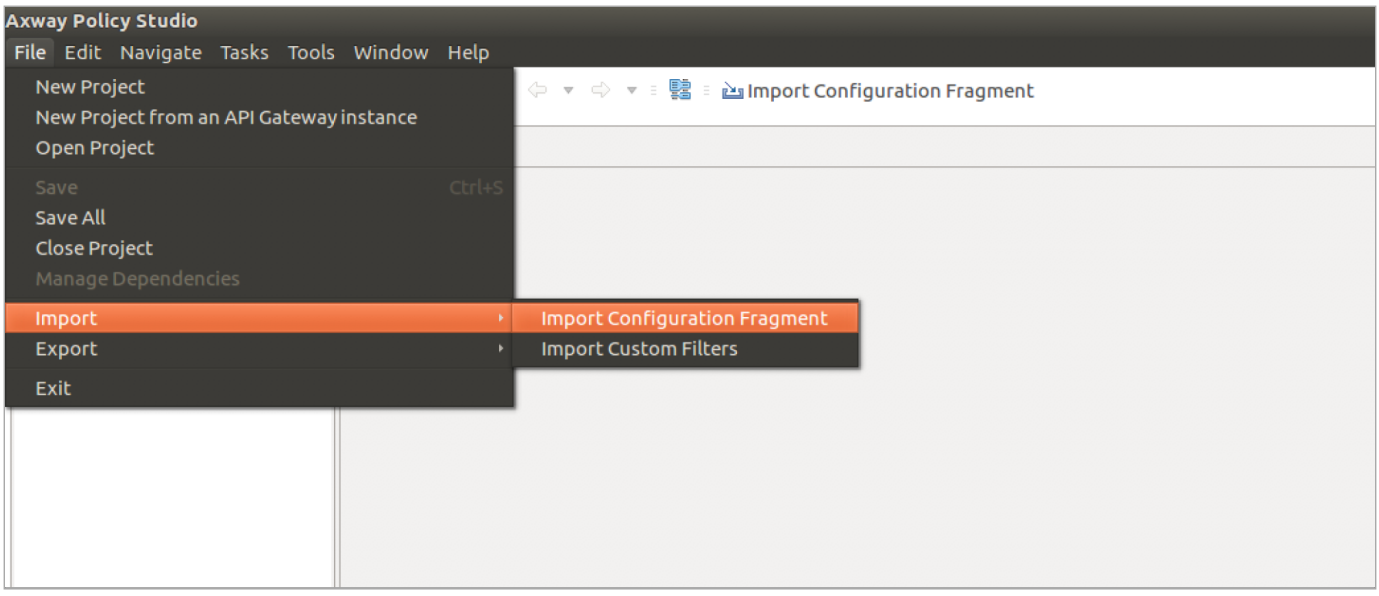


3. **Enter Host details, Username, and Password of the API Gateway to connect, and click Next >.**

Copyright © 2025 Ping Identity Corporation

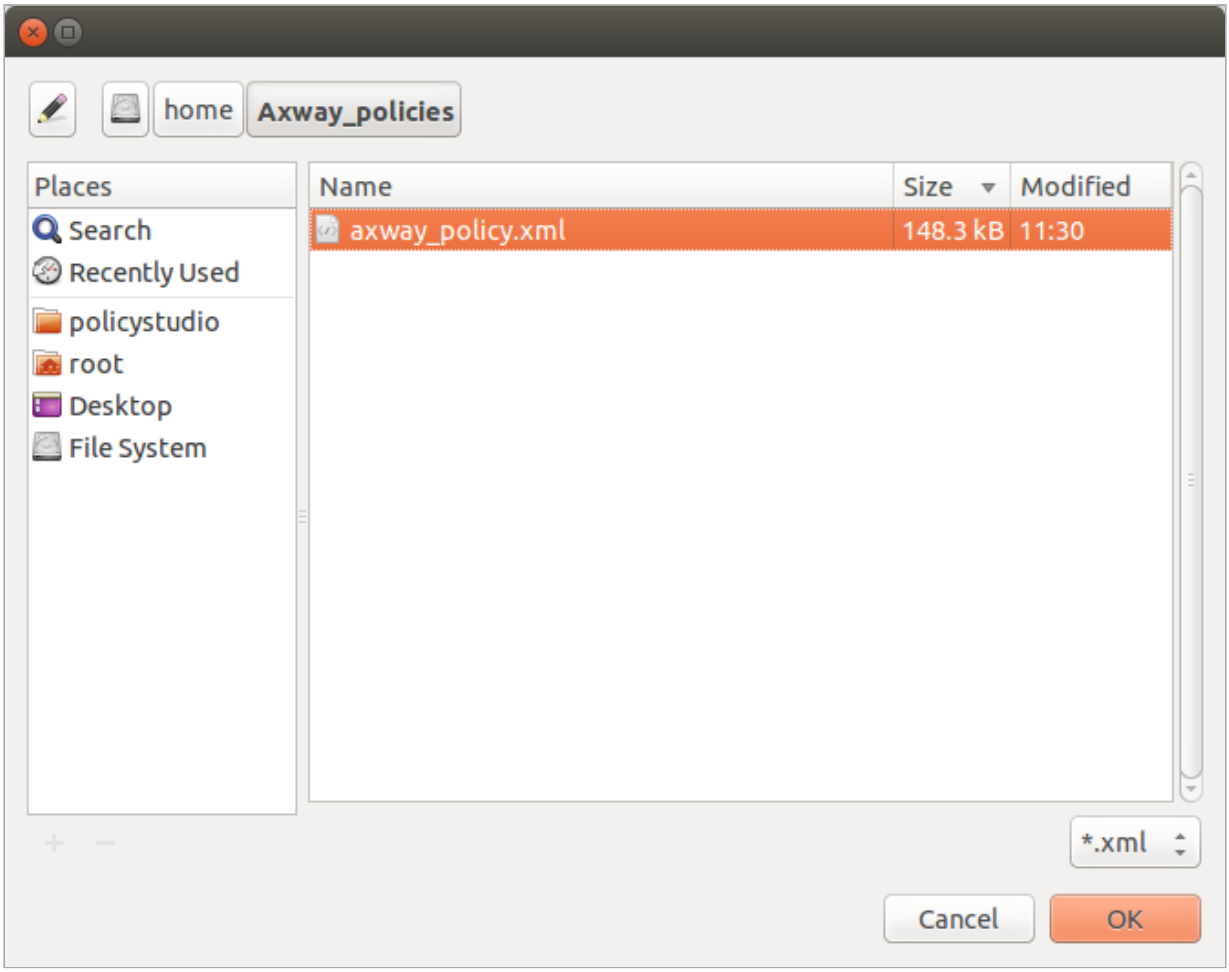


4. From the top menu, go to File → Import → Import Configuration Fragment.



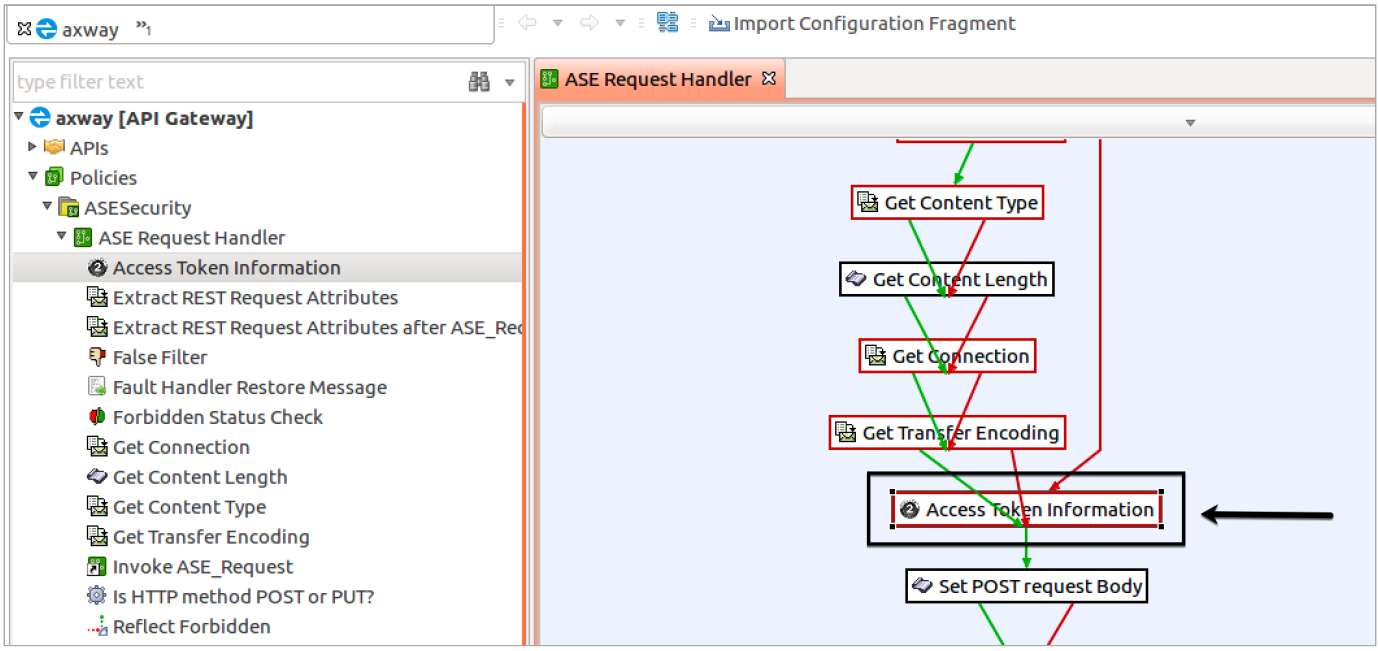
5. From the pop-up window, import the Axway policy from the directory where it was saved. Select the policy and click OK.

Copyright © 2025 Ping Identity Corporation

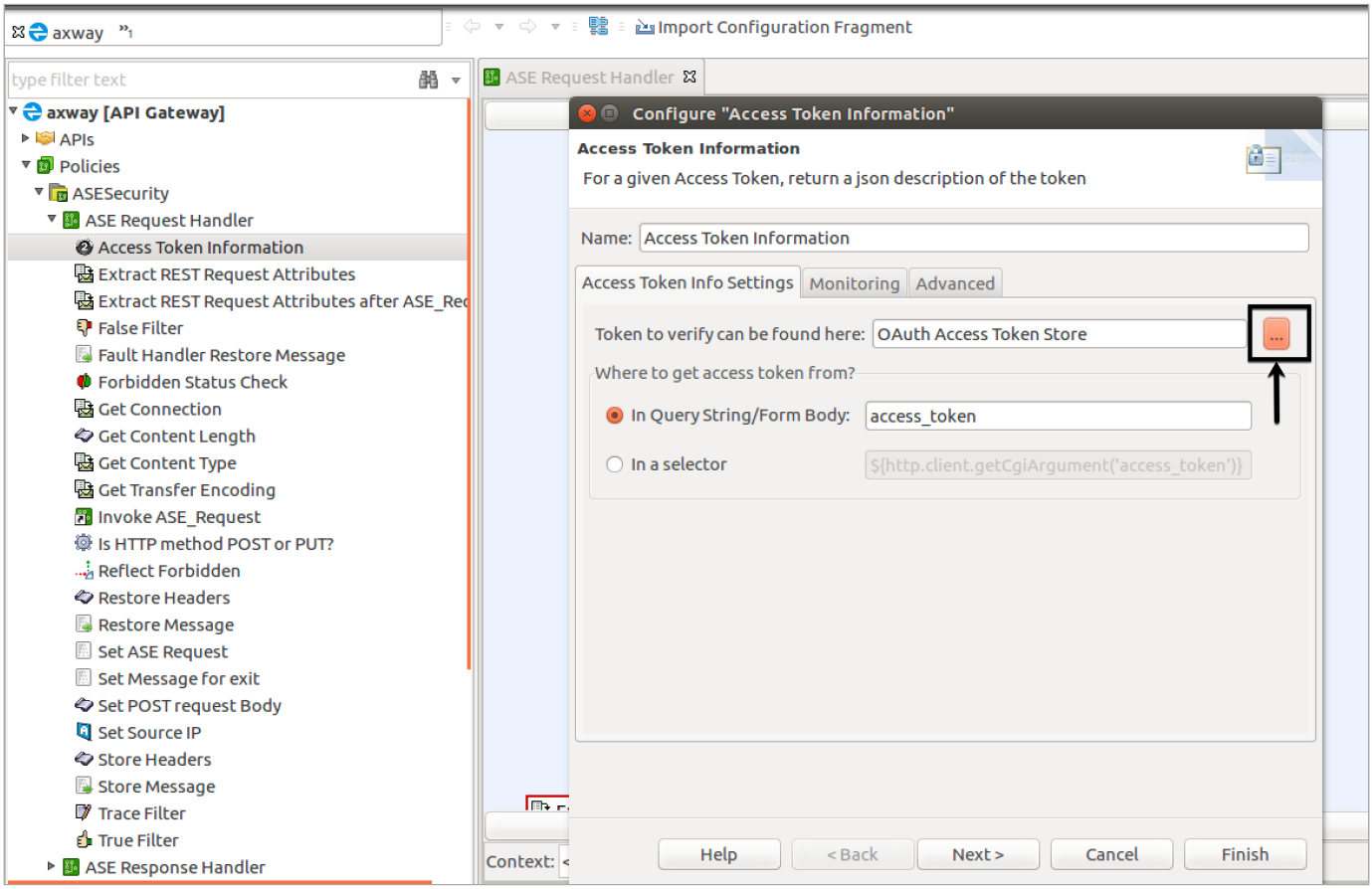


6. After the Axway Policy is imported, go to Policies → ASESecurity → ASE Request Handler → Access Token Information. Double click Access Token Information in the ASE Request Handler window.

Copyright © 2025 Ping Identity Corporation

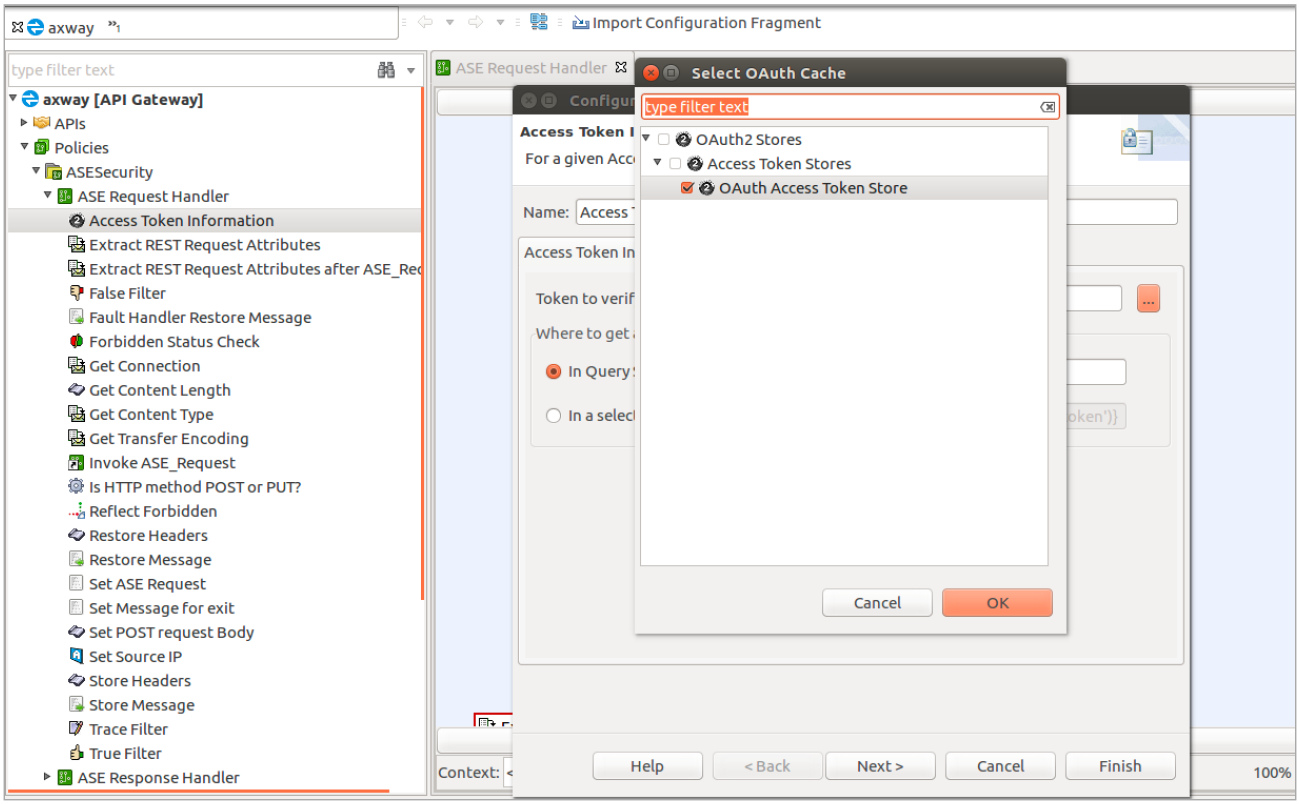


1. In the Configure "Access Token Information" pop-up window, enter your OAuth token store information and click the More Options button.

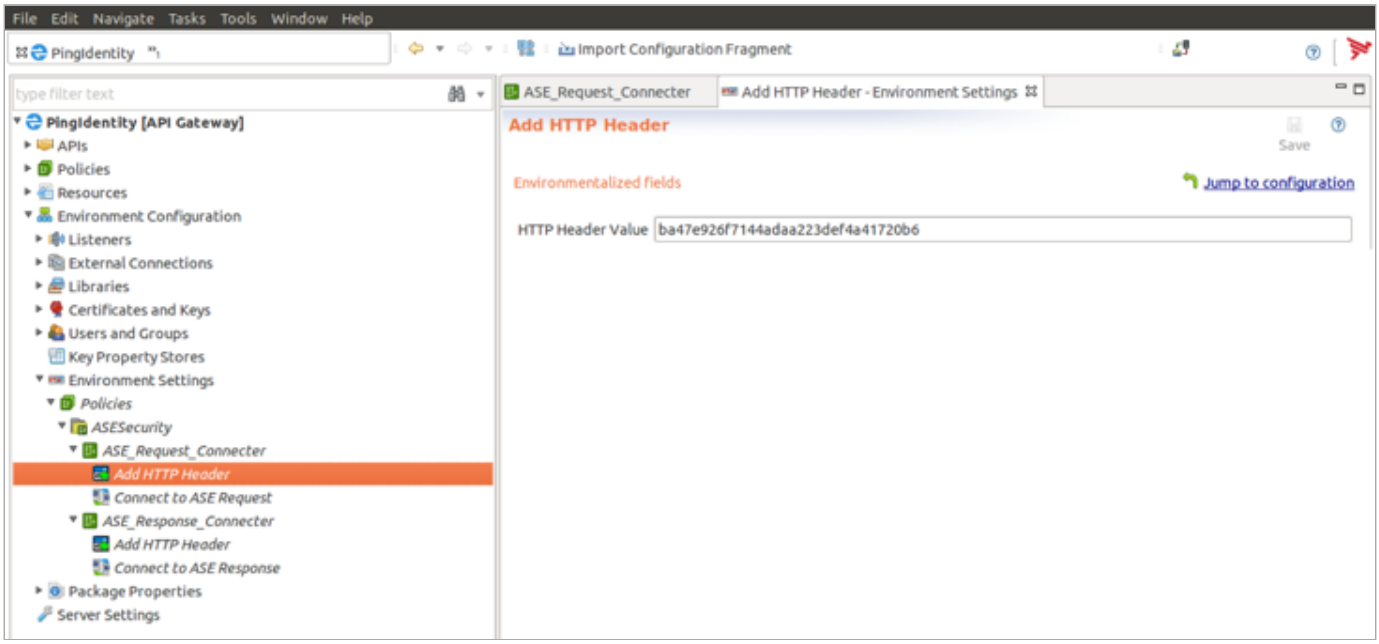


1. In the Select OAuth Cache pop-up window, select OAuth Access Token Store.

Copyright © 2025 Ping Identity Corporation

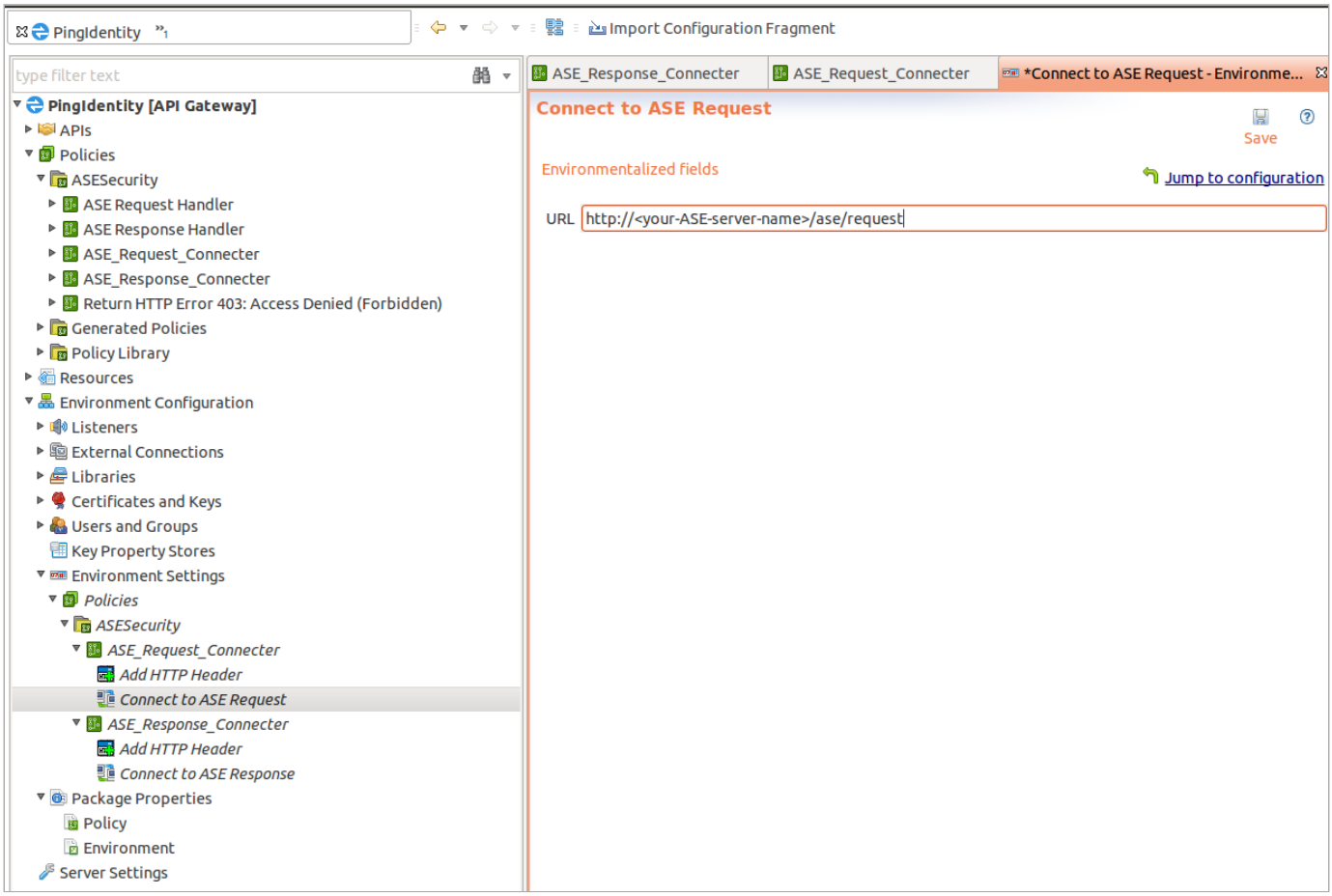


7. After the Axway policy is imported, click Environment Settings in the left-hand navigation and click Add HTTP Header. In the HTTP Header Value field, enter the ASE authentication token that was created in Preparing to configure the Axway API Gateway, step 6.



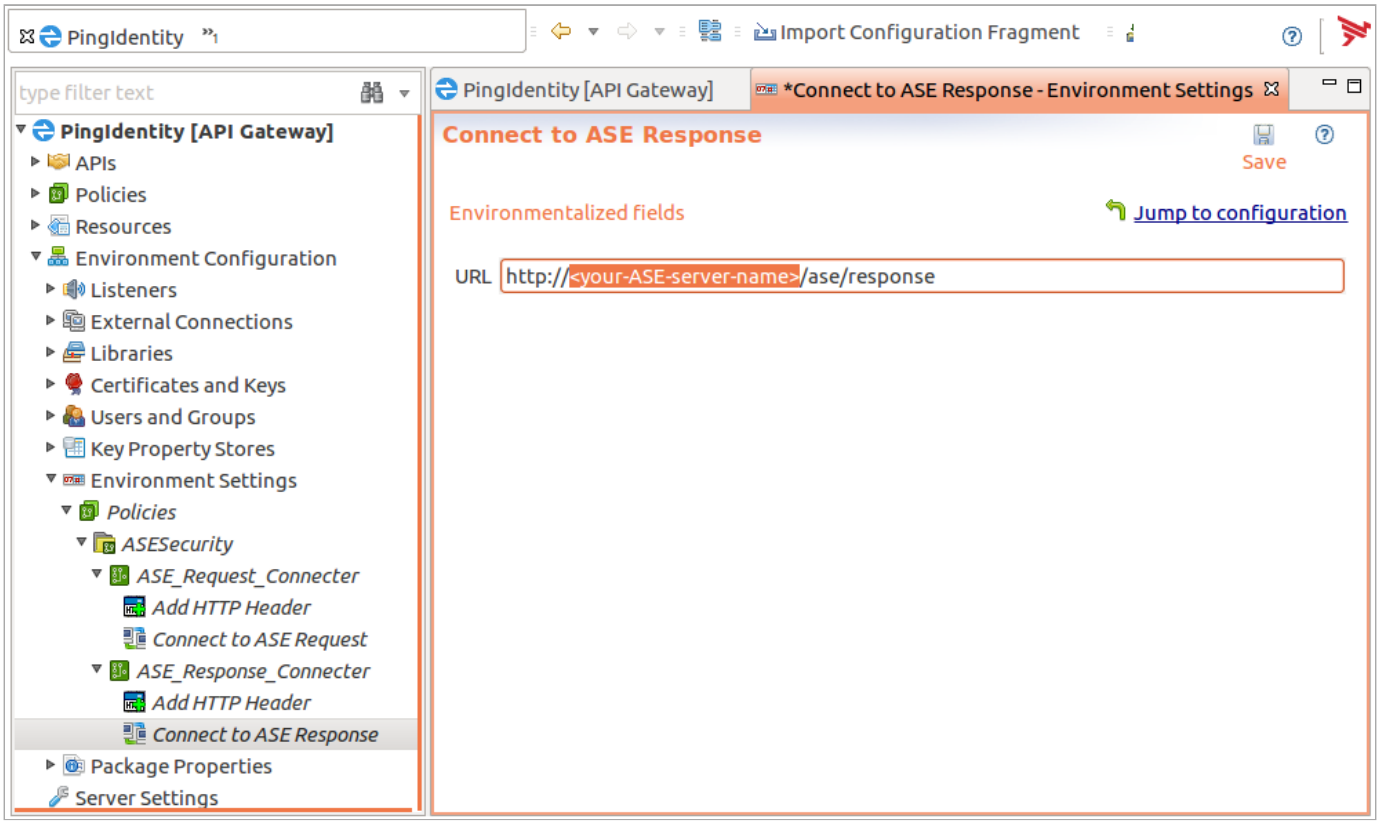
8. After the Axway policy is imported, go to Environment Settings → ASE_Request_Connector → Connect to ASE Request in the left-hand navigation. Enter the IP address or the hostname of your ASE in the URL field as shown below.

Copyright © 2025 Ping Identity Corporation



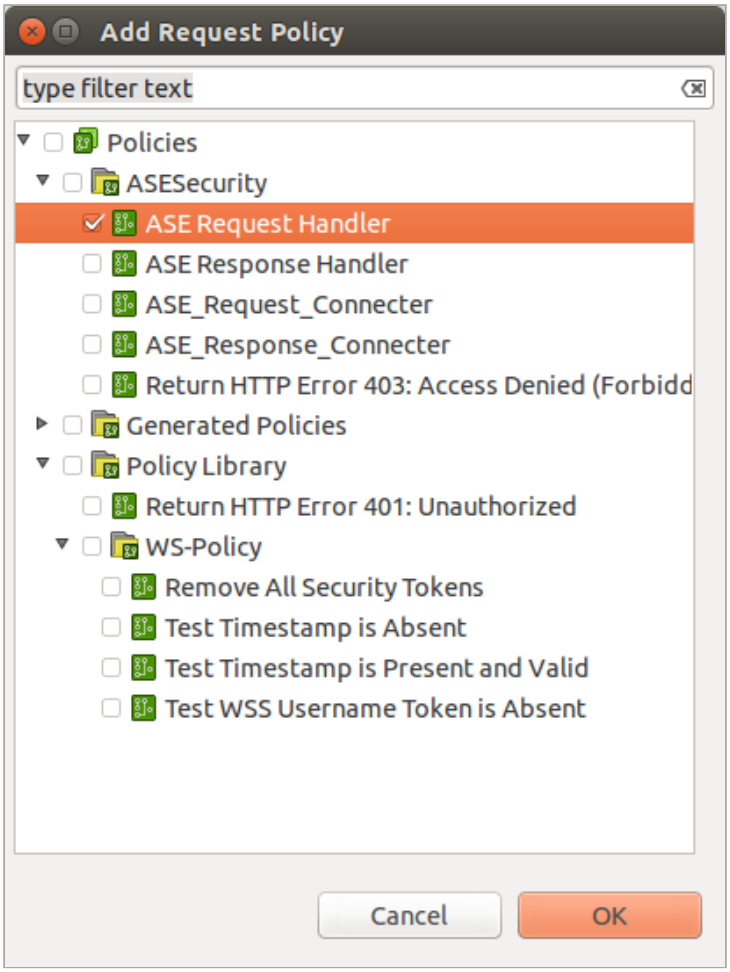
9. Go to Environment Settings → ASE_Response_Connector. → Connect to ASE Response in the left-hand navigation. Enter the IP address or the hostname of your ASE in the URL field as shown below.

Copyright © 2025 Ping Identity Corporation



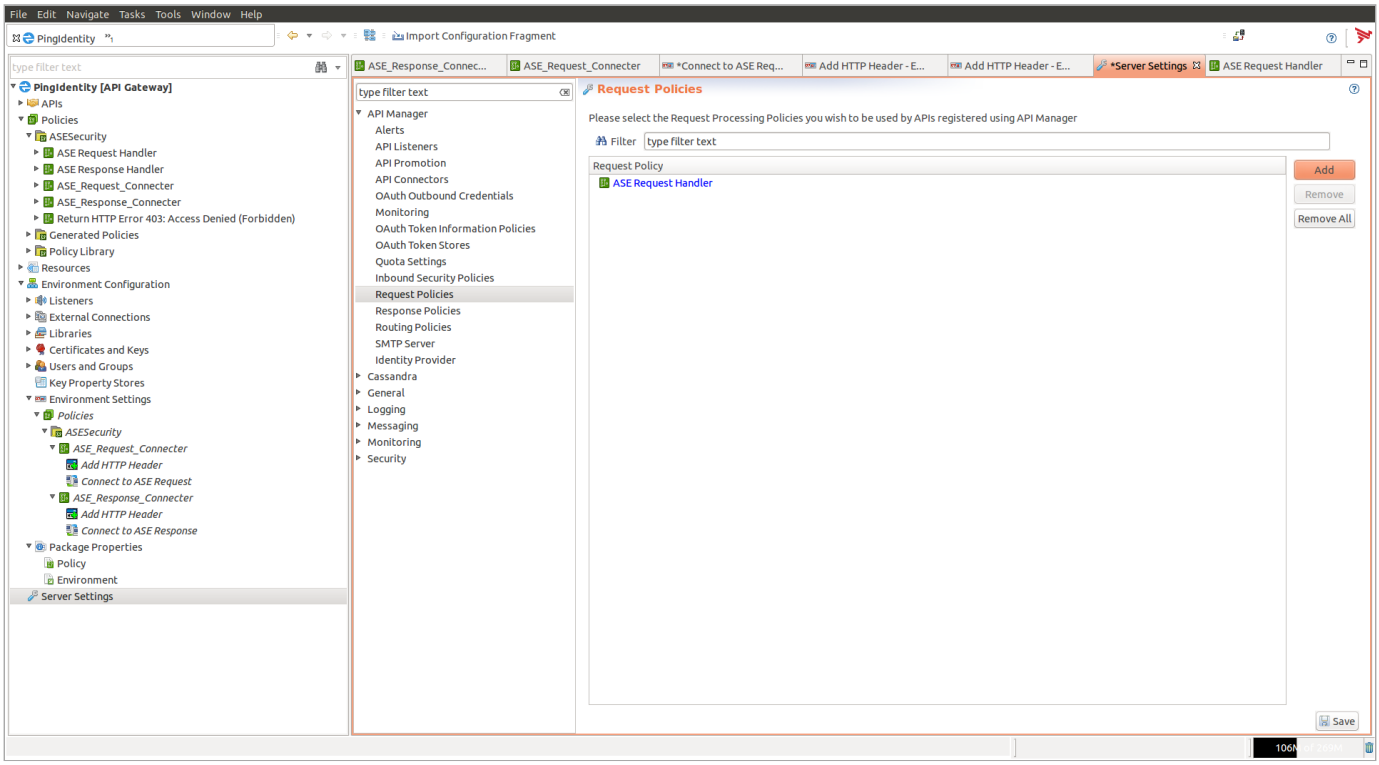
10. In the left navigation, click Server Settings.
11. In the Server Settings window, double-click Request Policies under API Manager.
12. In the Add Request Policy pop-up window, click the ASE Request Handler check box and click OK.

Copyright © 2025 Ping Identity Corporation



13. Click the Add button and then Save.

Copyright © 2025 Ping Identity Corporation



14. Repeat steps 10 and 11 for response policies.
15. To deploy the policies, click Deploy.

Optional: Configuring ASE persistent connections

You can optionally configure TCP keep-alive connections in the `ase.conf` file of API Security Enforcer (ASE).

About this task

The following is a snippet of `ase.conf` displaying the default `enable_sideband_keepalive` variable. The default value is set to `false`.

```
; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false
```

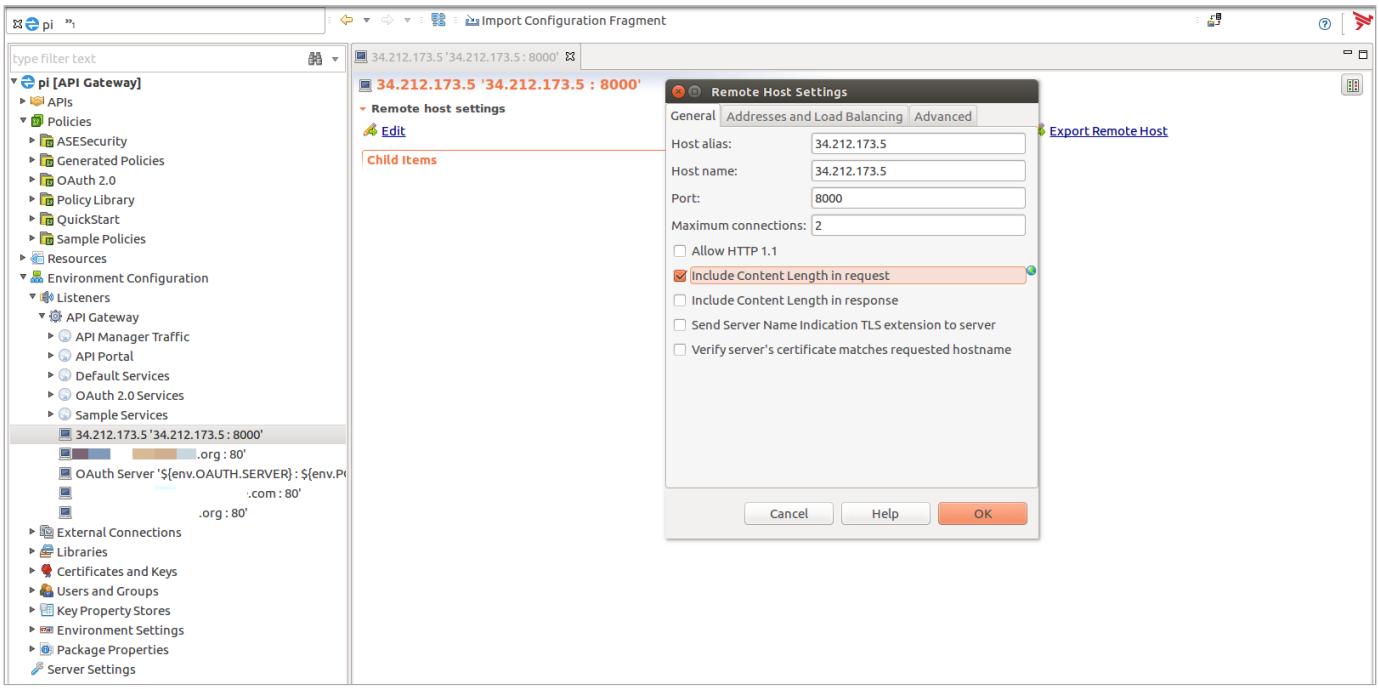
If this variable is set to `true`, then you must configure persistent connections in Axway Policy Studio:

Steps

1. In Axway Policy Studio, go to Environment Configuration → Listeners → API Gateway.
2. Click your ASE IP address in Sample Services.
3. In the Remote Host Settings pop-up window, clear the Allow HTTP 1.1check box.

Copyright © 2025 Ping Identity Corporation

4. Click the Include Content Length in request check box. Make sure all other check boxes are cleared.
5. Click OK and deploy the policy.



Configuring Axway API Manager

To deploy a PingIntelligence policy, configure Axway API Manager.

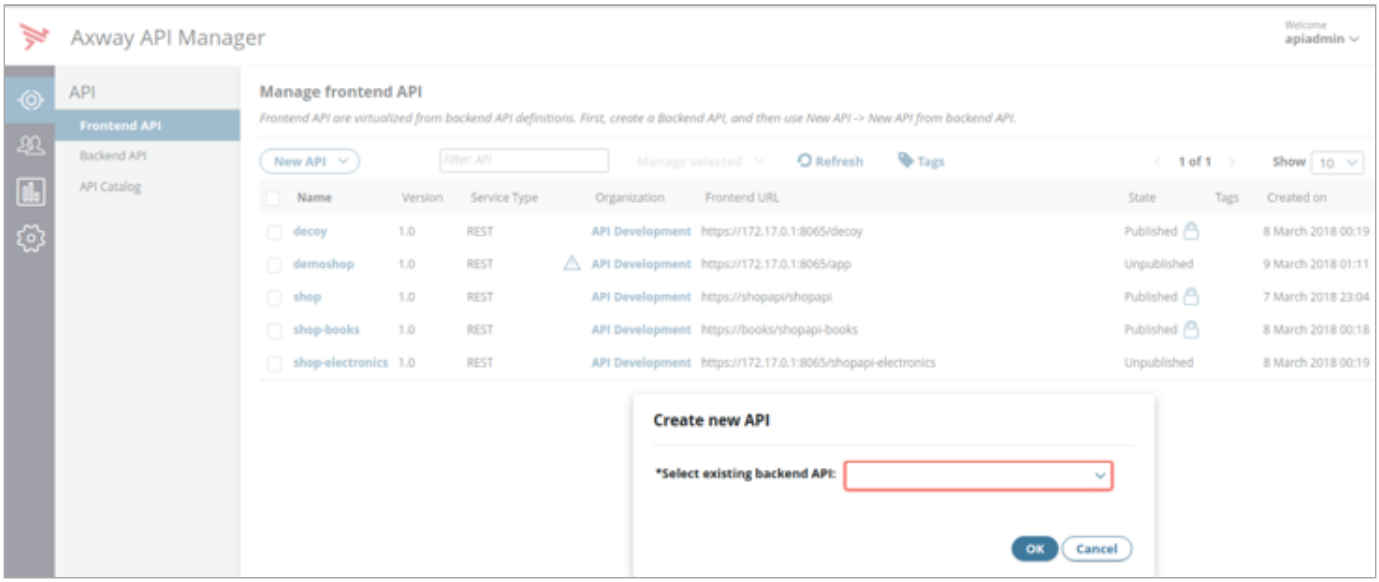
About this task

To configure Axway API Manager:

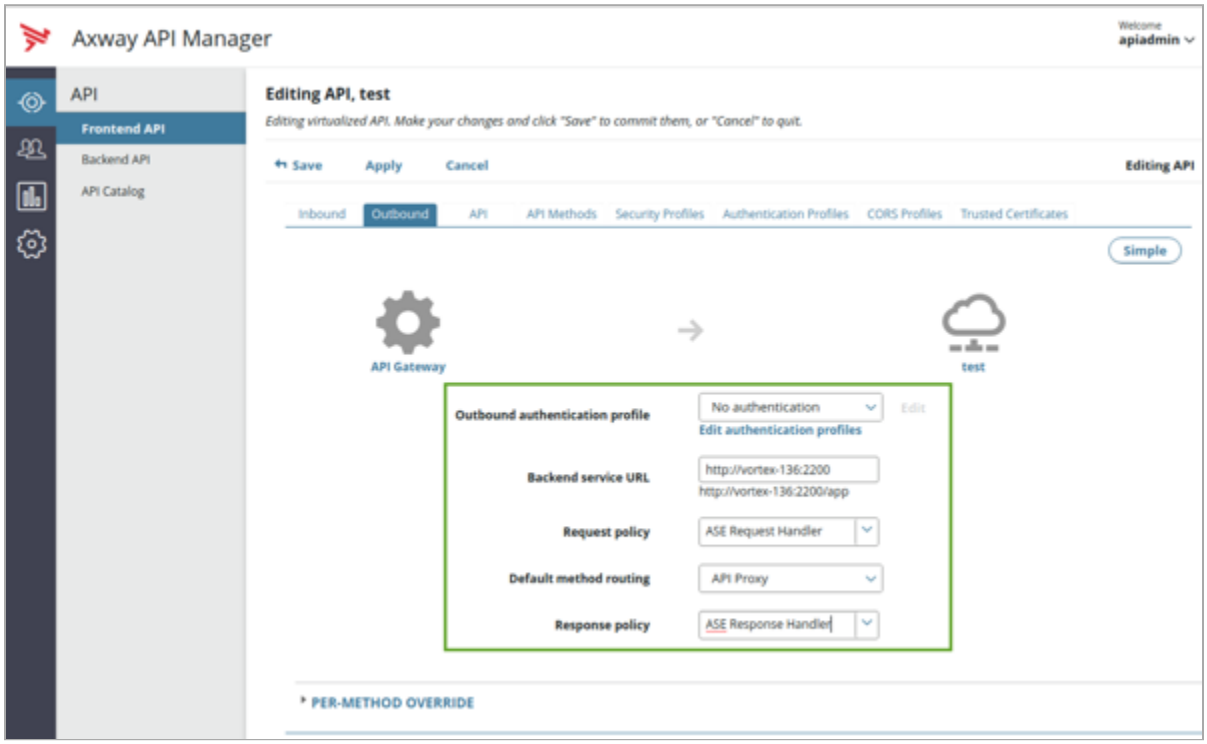
Steps

1. Sign on to Axway API Manager.
2. In Axway API Manager, click Frontend API and Create new API.

Copyright © 2025 Ping Identity Corporation



3. Click the Outbound tab and enter the Backend service URL (your backend application server) and Request policy details.

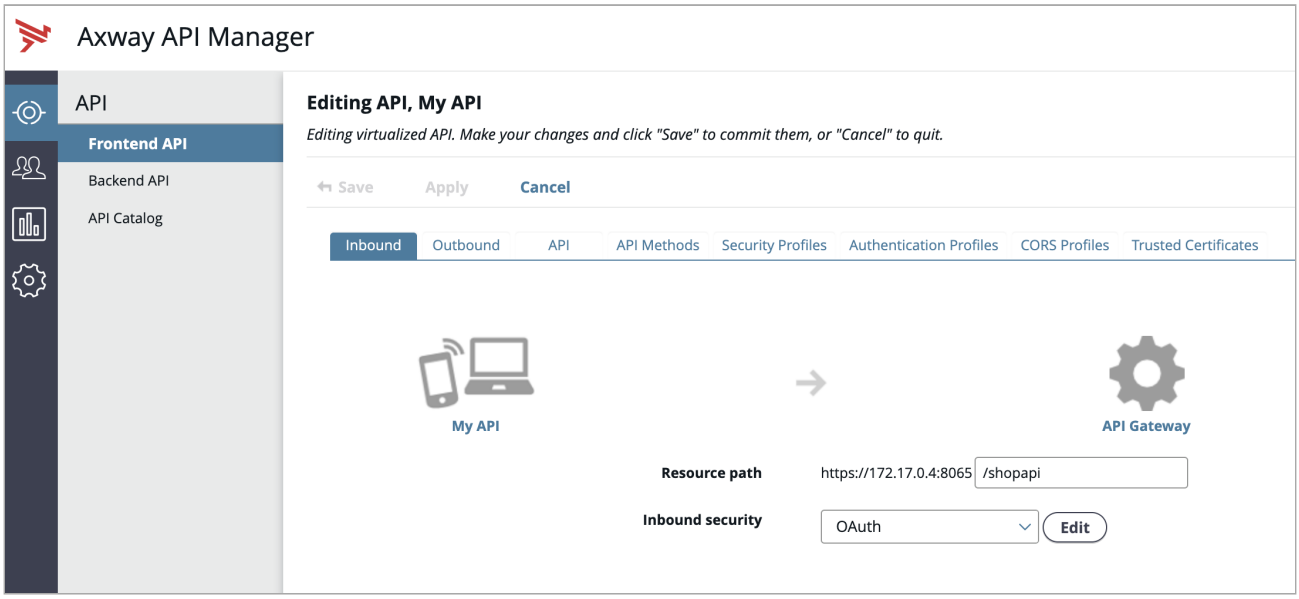


4. To capture OAuth token-based attacks.

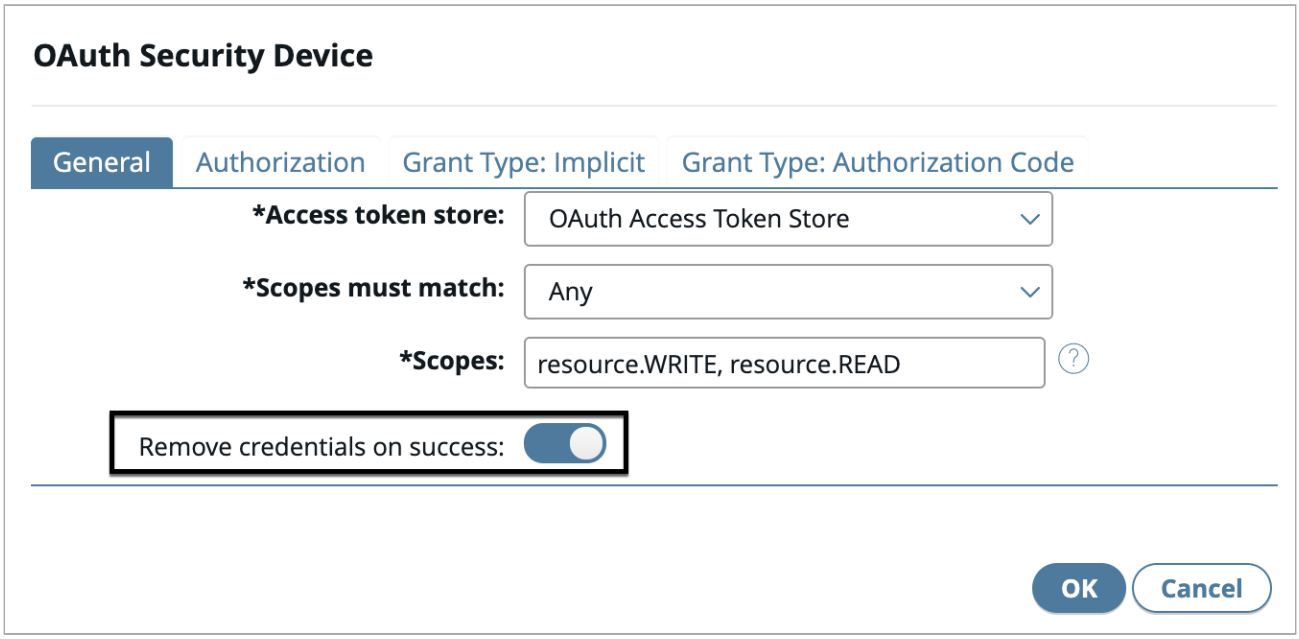
    1. In the API Manager, go to Frontend API → Inbound tab.

    2. From the Inbound security drop-down list, select OAuth and click Edit.

Copyright © 2025 Ping Identity Corporation



3. In the OAuth Security Device window, click the toggle to Remove credentials on success.



API discovery

PingIntelligence API discovery is a process to discover and report APIs from your API environment.

The discovered APIs are reported in the PingIntelligence Dashboard. APIs are discovered when a global API JavaScript Object Notation (JSON) is defined in API Security Enforcer (ASE).

For more information, see API discovery and configuration . You can edit the discovered API's JSON definition in theDashboard before adding them to ASE. For more information on editing and configuring API discovery, see Discovered APIs.

Copyright © 2025 Ping Identity Corporation

Configuring Axway API Manager for PingIntelligence Dashboard

The PingIntelligence Dashboard pulls the API definition from Axway API Manager and converts it to a JavaScript Object Notation (JSON) format compatible with ASE.

About this task

The Dashboard needs certain tags to be configured in Axway API Manager for it to import the normal and decoy API definitions.

To configure tags in Axway API Manager and tags for the decoy API:

Copyright © 2025 Ping Identity Corporation

API Manager

Configuring tags in API Manager
About this task

Tags are a medium to let ASE know which APIs from the API ecosystem need to be processed for monitoring and attack detection. Tags are also required for cookie and login Uniform Resource Locator (URL) parameters to be captured by the PingIntelligence Dashboard for adding to the ASE API JSON definition.

To tag APIs for artificial intelligence (AI) processing:

Steps

1. Configure the `ping_ai` tag for all the APIs for which you want the traffic to be processed using the AI engine.

   For example, if you have 10 APIs in your ecosystem and you want only traffic for 5 APIs to be processed using the AI engine, then apply the `ping_ai` tag on those 5 APIs.

   1. In the Axway API Manager, click Frontend API → API tab. In the API tab, navigate to the Tags section and add the following tag and value:

      - `ping_ai` – Set it to `true` if you want the traffic for API to be processed by PingIntelligence
      - `ping_blocking` – This parameter defines whether `enable_blocking` in the ASE API JSON is set to `true` or `false` when the PingIntelligence Dashboard fetches the API definition from Axway. The default value is `true`. If you want to disable blocking in ASE, set it to `false`.

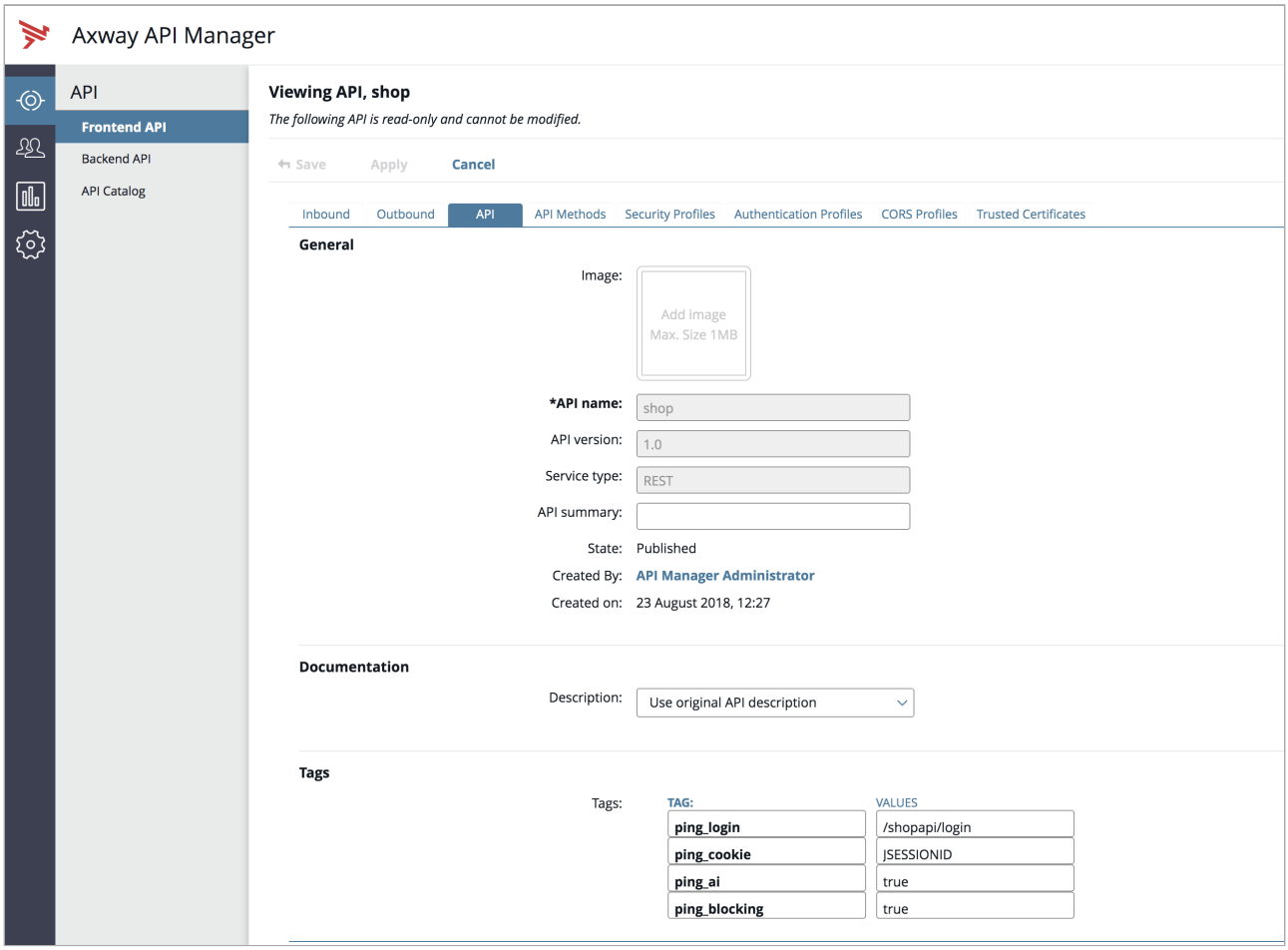
2. If your APIs use a cookie or login URL, then configure the following two tags and values for a cookie and login URL.

   1. In the Axway API Manager, go to Frontend API → API tab. In the API tab, navigate to Tags section and add the following tag and value:

      - `ping_cookie` – JSESSIONID
      - `ping_login` – yourAPI/login

      > **Note**
      >
      > If the API has API Key or OAuth2 token configured, the PingIntelligence Dashboard automatically learns it and adds it to the API JSON definition. You do not need to configure any tags for API Key and OAuth2 token.

Copyright © 2025 Ping Identity Corporation



Copyright © 2025 Ping Identity Corporation

Decoy APIs

Configuring tags for decoy APIs
About this task

You can configure decoy APIs in Axway API Manager. A decoy API is an API for which the traffic does not reach the backend API servers. The decoy API is deployed to gather information about potential threats that your API ecosystem may face. Traffic directed to a decoy API configured in Axway API Gateway is redirected to ASE, which functions as the backend server. ASE sends a preconfigured response, such as `200 OK`, for requests sent to a decoy API.

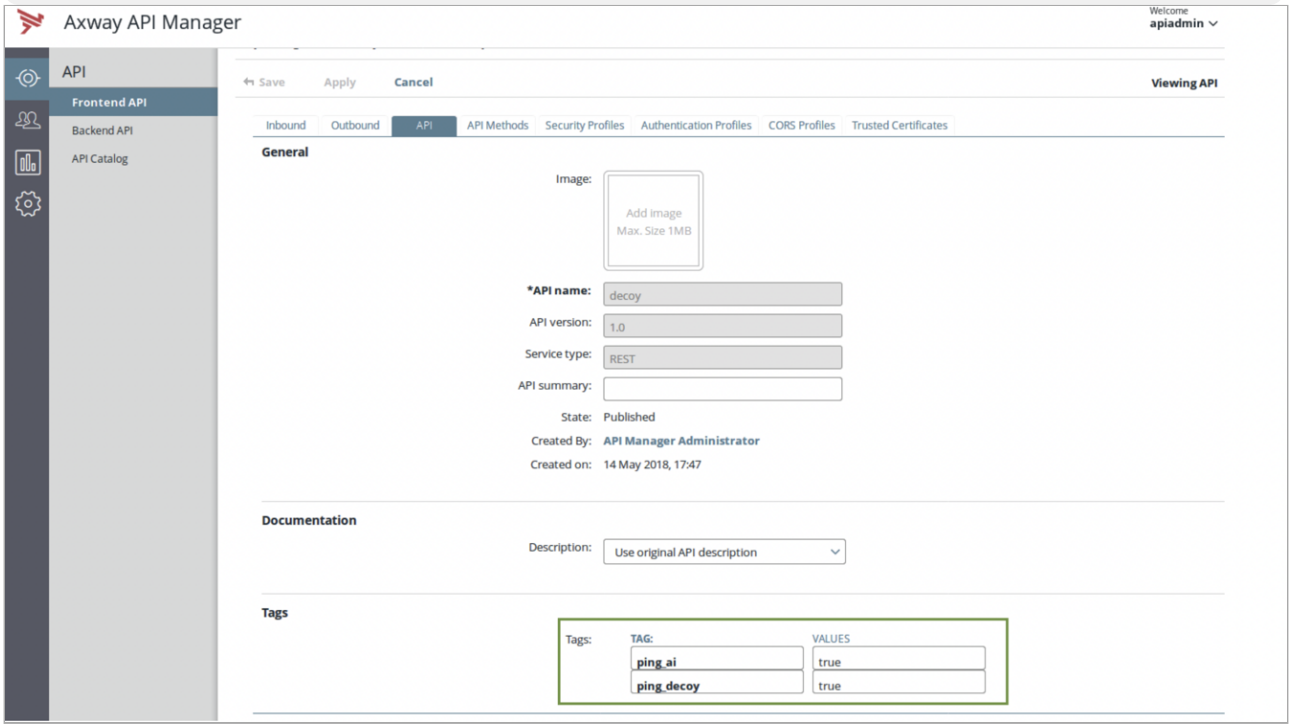
You need to configure the following TAGS and VALUES in the API tab for ** in Axway API Manager:

Steps

1. In Axway API Manager, go to Frontend API → API tab.
2. Configure the following tags and values:

```
ping_ai – true
```

```
ping_decoy – true
```



Example:

Copyright © 2025 Ping Identity Corporation

The converted API JSON will have the decoy section configured as highlighted in the following JSON file:

```
{
    "api_metadata": {
         "protocol": "https", "url": "/decoy", "hostname": "*",
        "cookie": "",
        "cookie_idle_timeout": "",
        "logout_api_enabled": false,
        "cookie_persistence_enabled": false,
        "oauth2_access_token": false,
        "apikey_qs": "",
        "apikey_header": "",
        "enable_blocking": true,
        "login_url": "",
        "api_mapping": {
            "internal_url": ""
        },
        "api_pattern_enforcement": {
            "protocol_allowed": "",
            "http_redirect": {
                "response_code": "",
                "response_def": "",
                "https_url": ""
            },
            "methods_allowed": [],
            "content_type_allowed": "",
            "error_code": "",
            "error_def": "",
            "error_message_body": ""
        },
        "flow_control": {
            "client_spike_threshold": "0/second",
            "server_connection_queueing": false
        },
        "api_memory_size": "64mb",
        "health_check": false,
        "health_check_interval": 60,
        "health_retry_count": 4,
        "health_url": "/",
        "server_ssl": false
        "servers": [],
         "decoy_config": \{ "decoy_enabled":true, "response_code": 200, "response_def": "OK",
"response_message": "OK", "decoy_subpaths": []
        }
    }
}
```

Configuring Axway XFF policies for decoy APIs

PingIntelligence provides an X-Forwarded-For (XFF) policy for your decoy APIs.

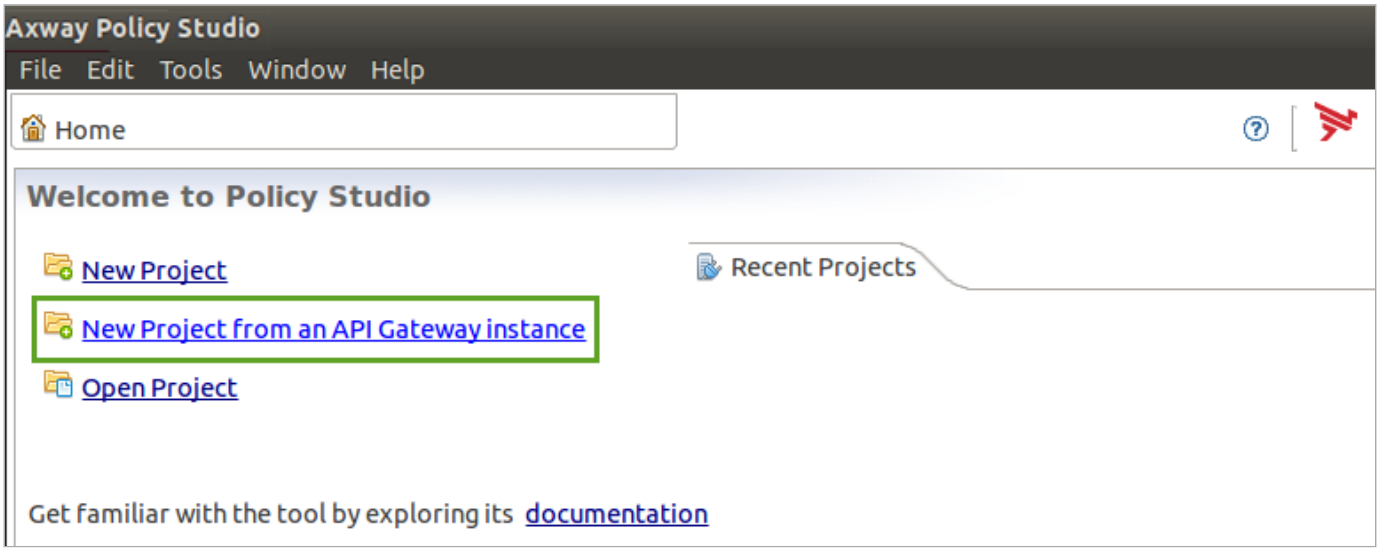
About this task

Copyright © 2025 Ping Identity Corporation

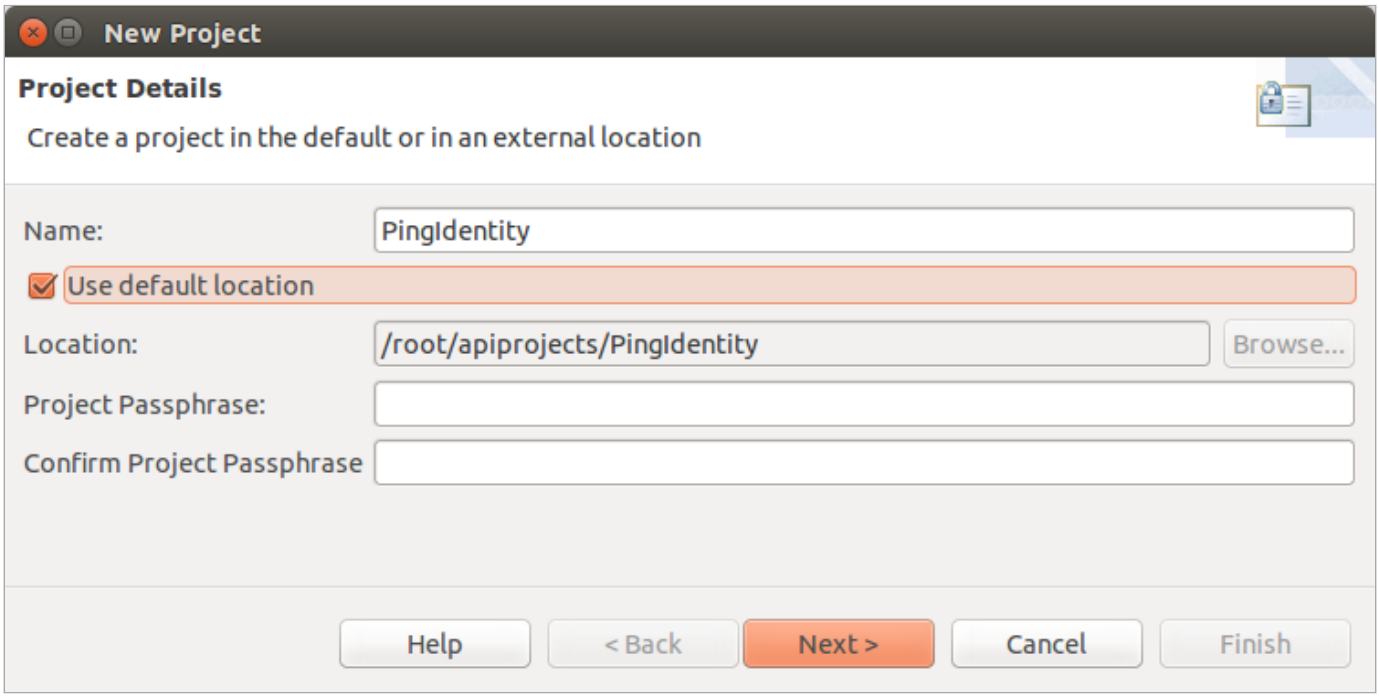
The XFF policy adds an `X-Forwarded-For` to the backend only if it is not present in the original incoming request. If the `X-Forwarded-For` header is already present in the incoming request, the policy takes no action

Steps

1. Launch Axway Policy Studio and click New Project from an API Gateway instance.

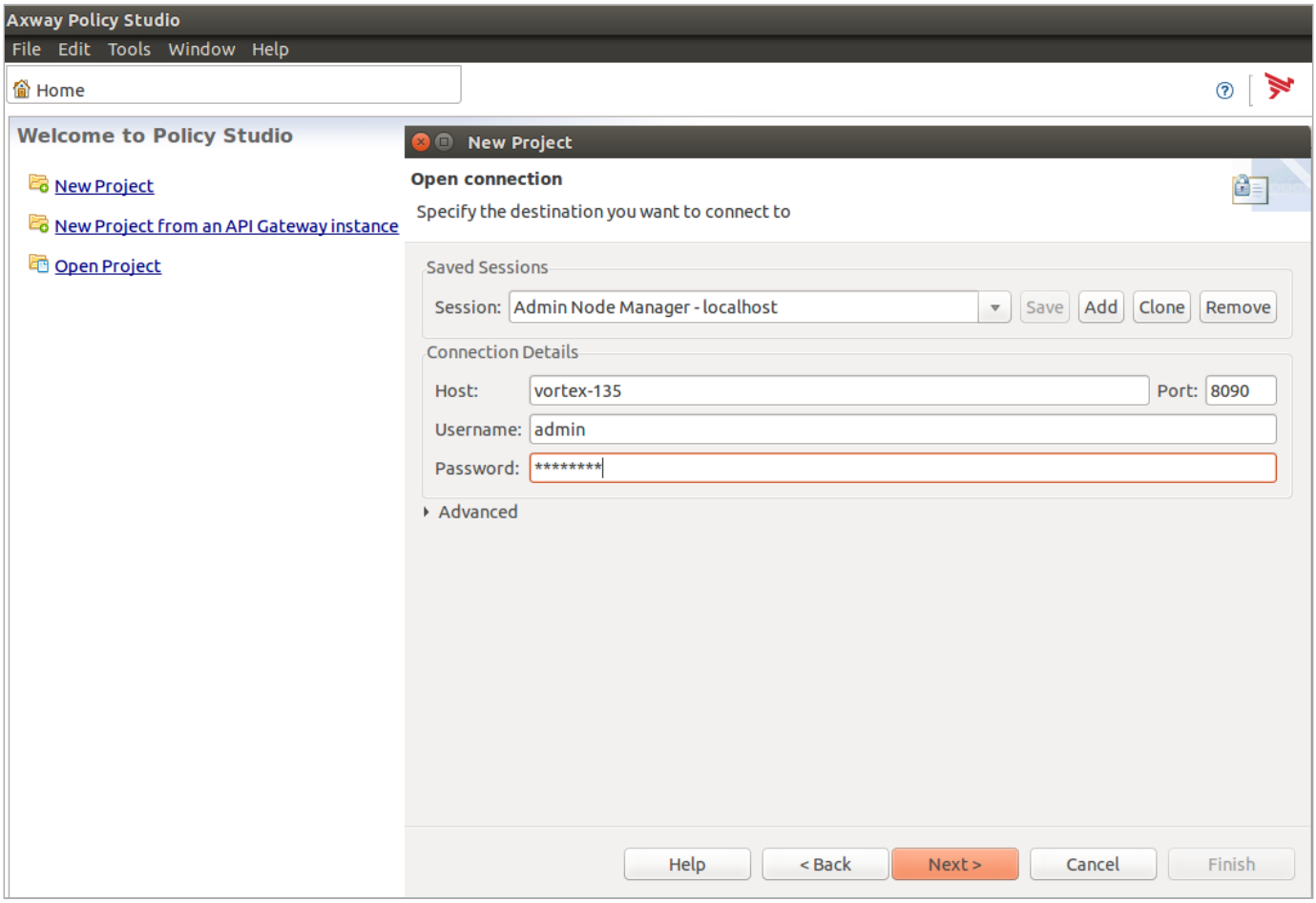


2. In the New Project pop-up window, enter the details and click Next >.

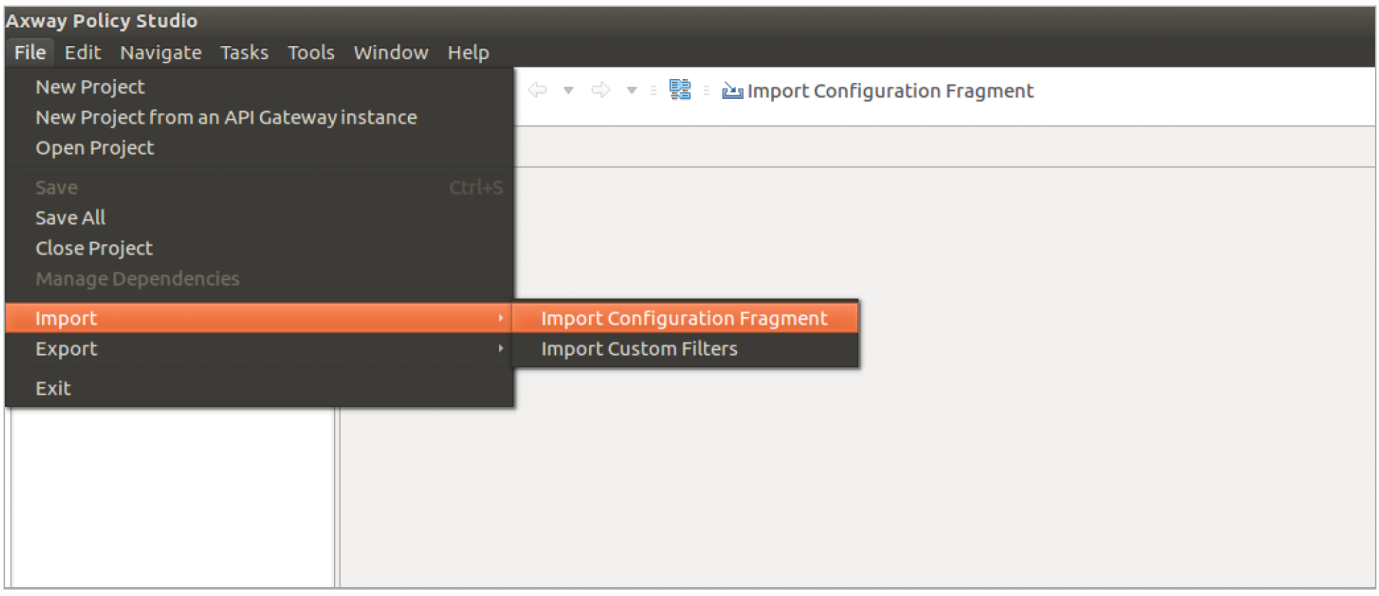


3. Enter Host details, Username, and Password of the API Gateway to connect, and click Next >.

Copyright © 2025 Ping Identity Corporation

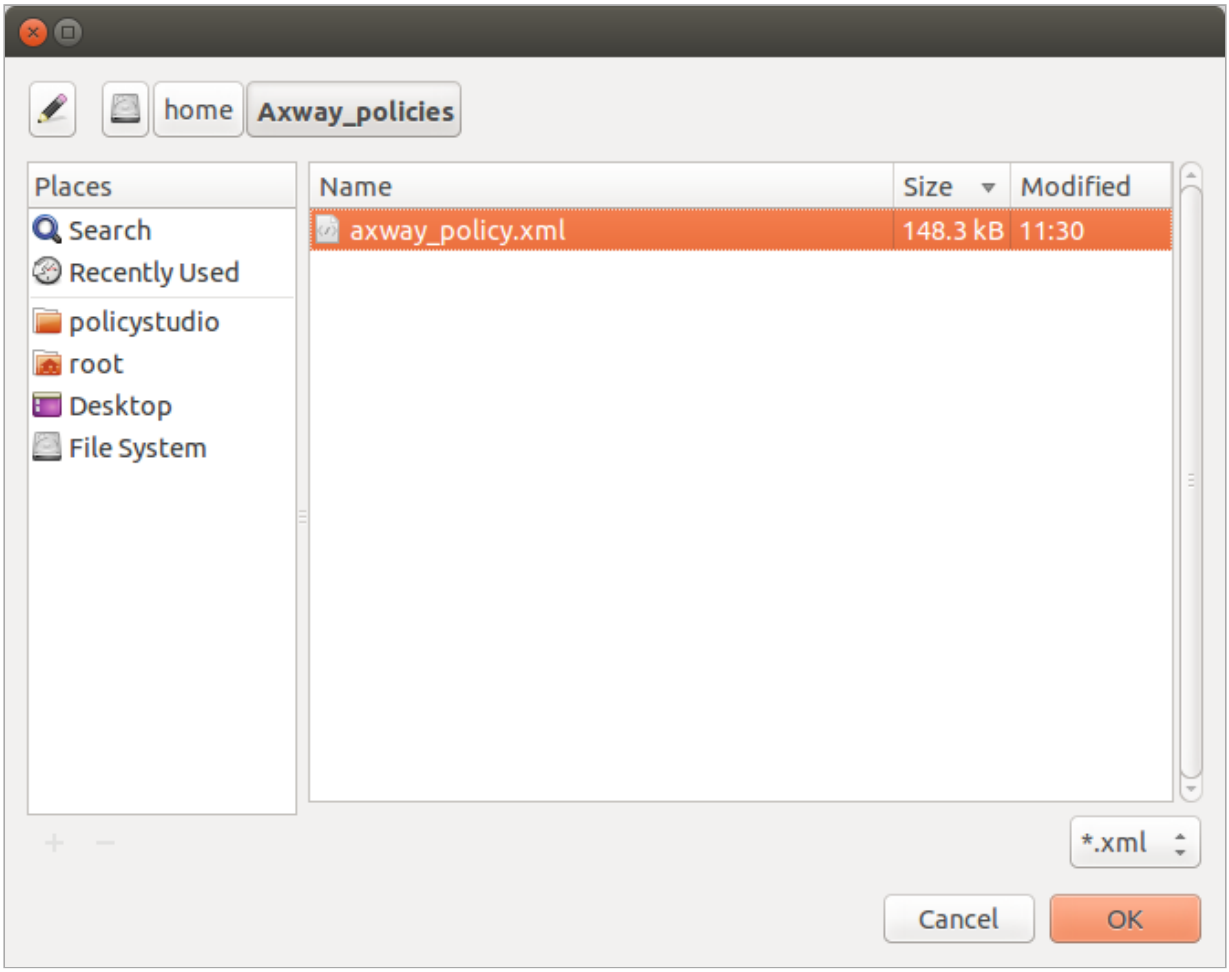


4. From the top menu, go to File → Import → Import Configuration Fragment.

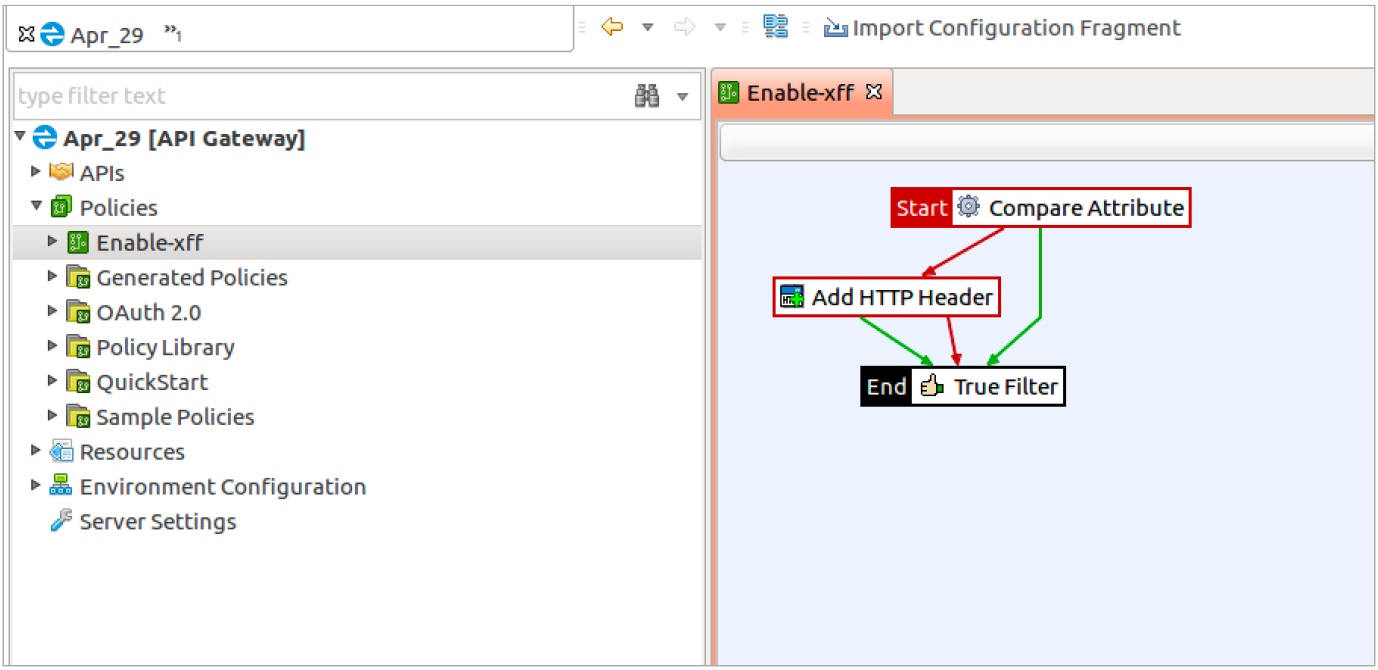


5. From the pop-up window, import the Axway policy from the directory where it was saved. Select the policy and click OK.

Copyright © 2025 Ping Identity Corporation



6. After importing the Axway policy, deploy the XFF policy



Copyright © 2025 Ping Identity Corporation

OAuth2 tokens and API keys

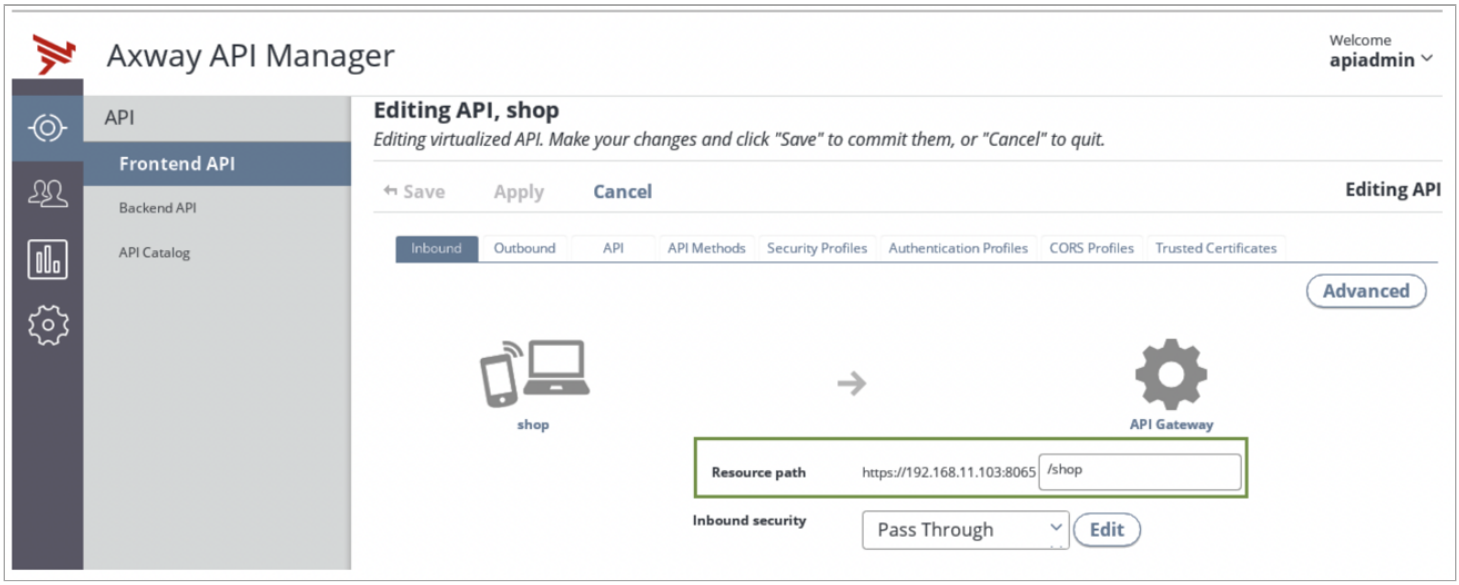
If you have configured the API key in Request Header or in Query String, the PingIntelligence Dashboard reads and converts these values to `apikey_qs` or `apikey_header` values in the API Security Enforcer (ASE) API JavaScript Object Notation (JSON).

The PingIntelligence artificial intelligence (AI) engine considers API key values only in request headers or the query string.

Similarly, if you have configured an OAuth2 token, the PingIntelligence Dashboard marks the value of `oauth2_access_token` as `true` in the ASE API JSON.

> **Note**
>
> You do not need to configure any tags for API keys or an OAuth2 token.

The following API JSON file shows the converted parameters. The `protocol`, `url`, and `hostname` values are picked from the values that you configure in Resource path when you create the Frontend API.



Copyright © 2025 Ping Identity Corporation

```
{
    "api_metadata": {
         "protocol": "https", "url": "/shop", "hostname": "192.168.11.103", "cookie": "JSESSIONID",
        "cookie_idle_timeout": "",
        "logout_api_enabled": false,
        "cookie_persistence_enabled": false,
        "oauth2_access_token":true, "apikey_qs": "KeyId", "apikey_header": "",
        "enable_blocking": true,
         "login_url": "/shop/login",
        "api_mapping": {
            "internal_url": ""
        },
        "api_pattern_enforcement": {
            "protocol_allowed": "",
            "http_redirect": {
                "response_code": "",
                "response_def": "",
                "https_url": ""
            },
            "methods_allowed": [],
            "content_type_allowed": "",
            "error_code": "",
            "error_def": "",
            "error_message_body": ""
        },
        "flow_control": {
            "client_spike_threshold": "0/second",
            "server_connection_queueing": false
        },
        "api_memory_size": "64mb",
        "health_check": false,
        "health_check_interval": 60,
        "health_retry_count": 4,
        "health_url": "/",
        "server_ssl": false
        "servers": [],
        "decoy_config": {
            "decoy_enabled": false,
            "response_code": 200,
            "response_def": "",
            "response_message": "",
            "decoy_subpaths": []
        }
    }
}
```

Azure APIM sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with Azure API Manager (APIM).

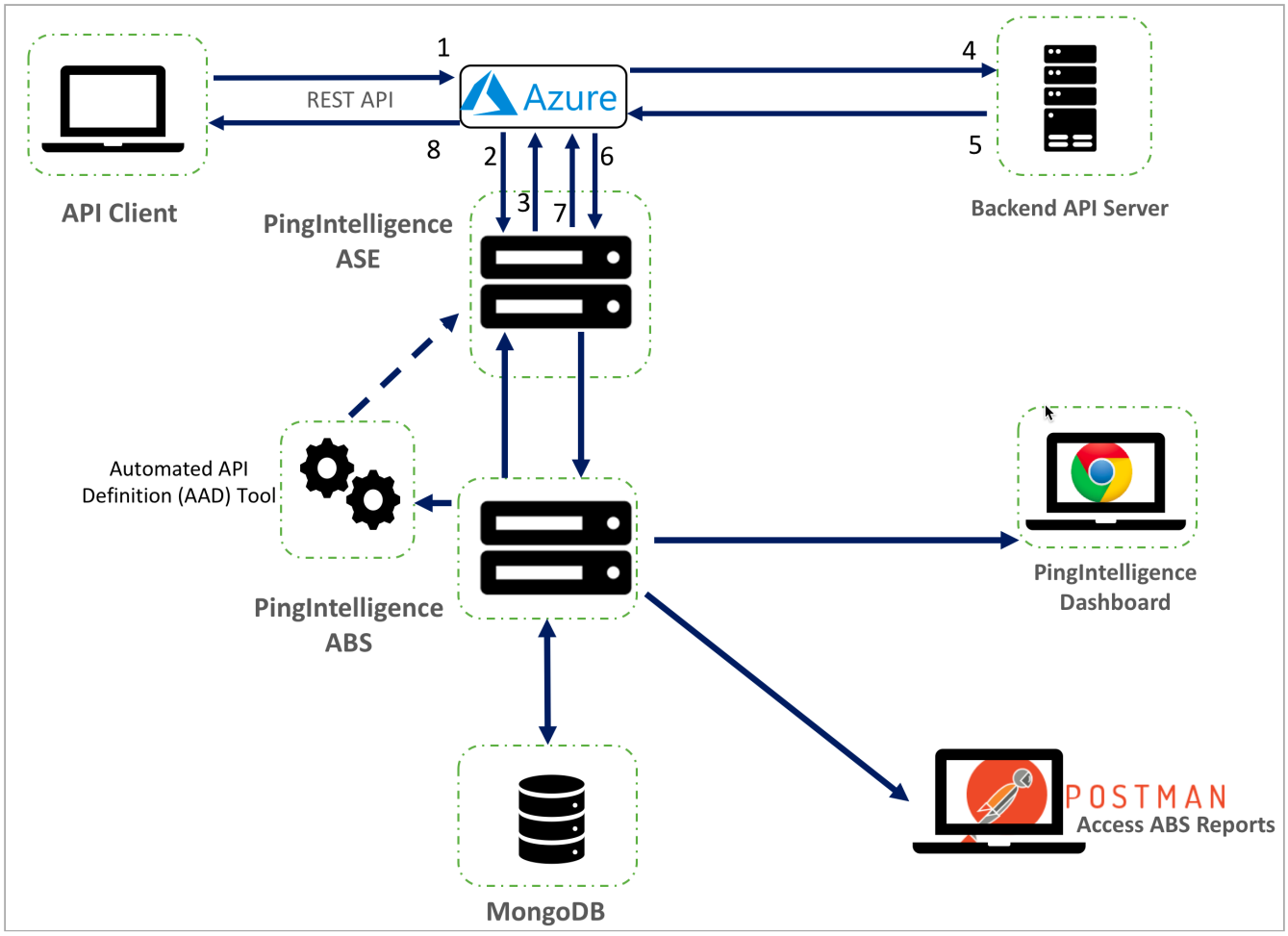
A PingIntelligence policy is installed in APIM and passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking. The PingIntelligence policy for Azure also supports detecting attacks based on the username.

Copyright © 2025 Ping Identity Corporation

The APIM PingIntelligence policy works in the following two configurable modes:

- Asynchronous mode: When the PingIntelligence policy is configured in the Asynchronous mode, APIM does not wait for a response from PingIntelligence API Security Enforcer (ASE) before sending the API client request to the backend API server. In this mode, PingIntelligence deployment passively logs the API request and response. It performs detailed API activity reporting and attack detection without blocking of attacks.
- Synchronous mode: When the PingIntelligence policy is configured in the Synchronous mode, Azure API gateway waits for a response from PingIntelligence ASE before sending the request to the backend API server or blocking it. In this mode, PingIntelligence actively logs and responds to the API requests and response. It performs detailed API activity reporting with attack detection and blocking of attacks.

The following diagram shows the logical setup of PingIntelligence ASE and Azure:



Below is the traffic flow through the Azure and PingIntelligence for APIs components.

1. Client sends an incoming request to APIM.
2. APIM makes an API call to send the request metadata to ASE.
3. ASE checks the request against a registered set of APIs and looks up the origin Internet Protocol (IP), cookie, OAuth2 token, or API key on the PingIntelligence AI engine-generated deny list. If all checks pass, ASE returns a `200-OK` response to APIM. If not, a different response code is sent to APIM. The request information is also logged by ASE and sent to the AI engine for processing.

Copyright © 2025 Ping Identity Corporation

4. If APIM receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, if it receives a `403-forbidden` response, the APIM blocks the client when blocking is enabled for the API.
5. The response from the backend server is received by APIM.
6. APIM makes a second API call to pass the response information to ASE which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to Azure.
8. APIM sends the response received from the backend server to the client.

Preparing to deploy the PingIntelligence policy on APIM

Complete the following prerequisites before deploying the PingIntelligence policy on API Manager (APIM):

About this task

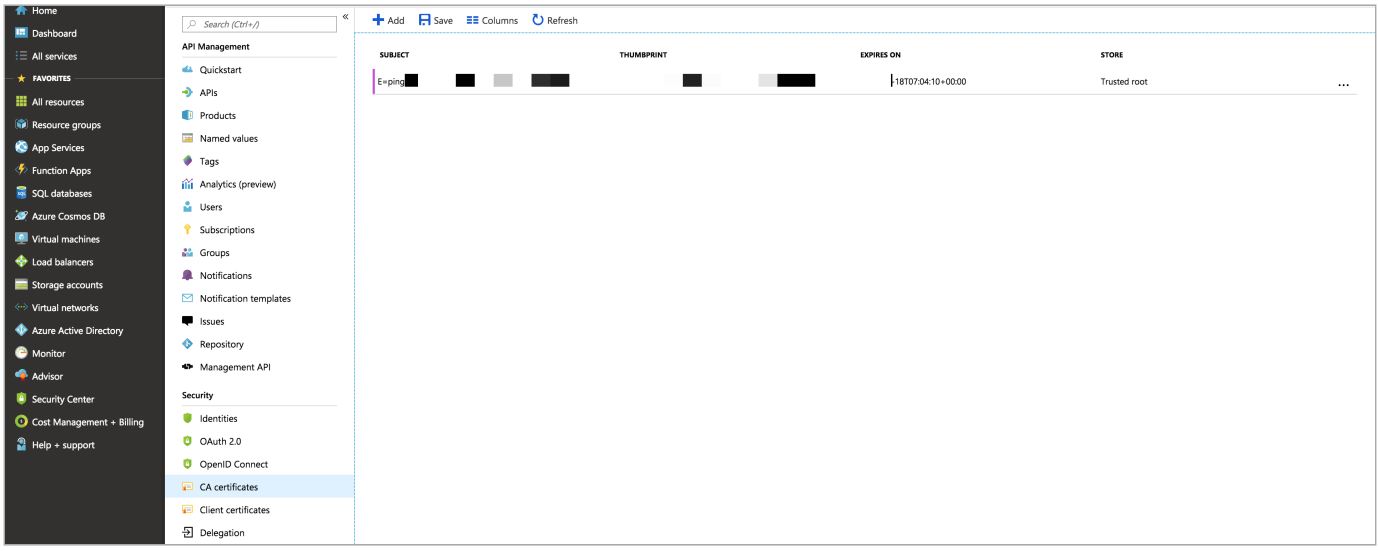
Before deploying the PingIntelligence policy on APIM:

Steps

1. Confirm that the Azure APIM Service is available.

   The PingIntelligence policy supports Azure APIM Q2CY2020 version. If you are using any other version, contact Ping Identity support.

2. Confirm that the APIs to which you want to apply the PingIntelligence policy are available.
3. To use the API Security Enforcer (ASE) self-signed certificate, configure the CA certificate from the Security → CA certificates the section.



4. Select one of the following four levels to apply the PingIntelligence policy:
   - For all the APIs
   - For a group of APIs, that is, at the product level

Copyright © 2025 Ping Identity Corporation

- For individual APIs
- For a specific operation in the API

5. Install and configure the PingIntelligence software.

   Refer to the PingIntelligence deployment guide for your environment type.

6. Verify that ASE is in `sideband` mode by running the following ASE command:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

   ```
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : disabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.61 MB, free 102.39 MB
   google pubsub           : disabled
   log level               : debug
   timezone                : local (UTC)
   ```

   Troubleshooting:

   If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

7. For a secure communication between APIM and ASE, enable sideband authentication by entering the following ASE command:

   ```
   # ./bin/cli.sh enable_sideband_authentication -u admin –p
   ```

8. A token is required for APIM to authenticate with ASE. To generate the token in ASE, enter the following ASE command and save the generated authentication token for further use:

   ```
   # ./bin/cli.sh -u admin -p admin create_sideband_token
   ```

Deploying the PingIntelligence policy

PingIntelligence provides an XML Extensible Markup Language (XML) policy file to integrate PingIntelligence and Azure API Management Service.

About this task

Copyright © 2025 Ping Identity Corporation

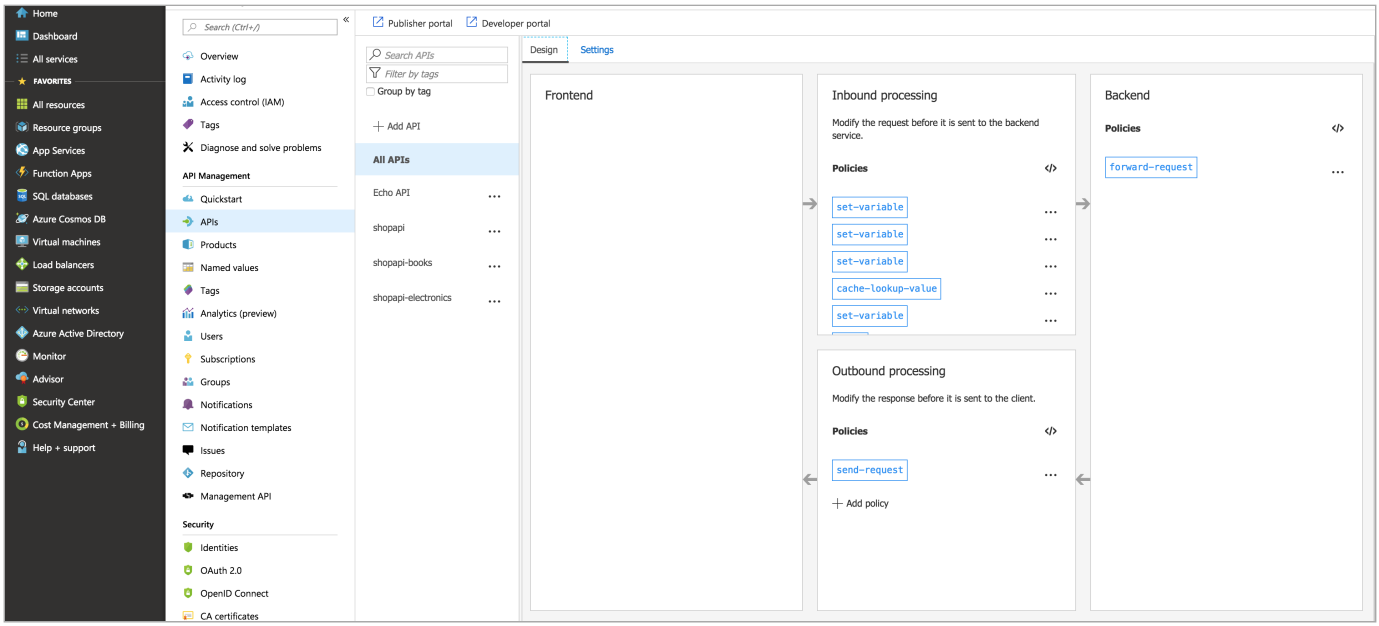
This policy can be applied at an individual API level, for all the APIs, to a group of APIs, or for an operation of an API. PingIntelligence recommends that the PingIntelligence policy be the first policy in the Azure policy XML. This ensures that all the traffic is captured by ASE and sent to the PingIntelligence artificial intelligence (AI) engine for analysis.

To deploy the PingIntelligence policy:

Steps

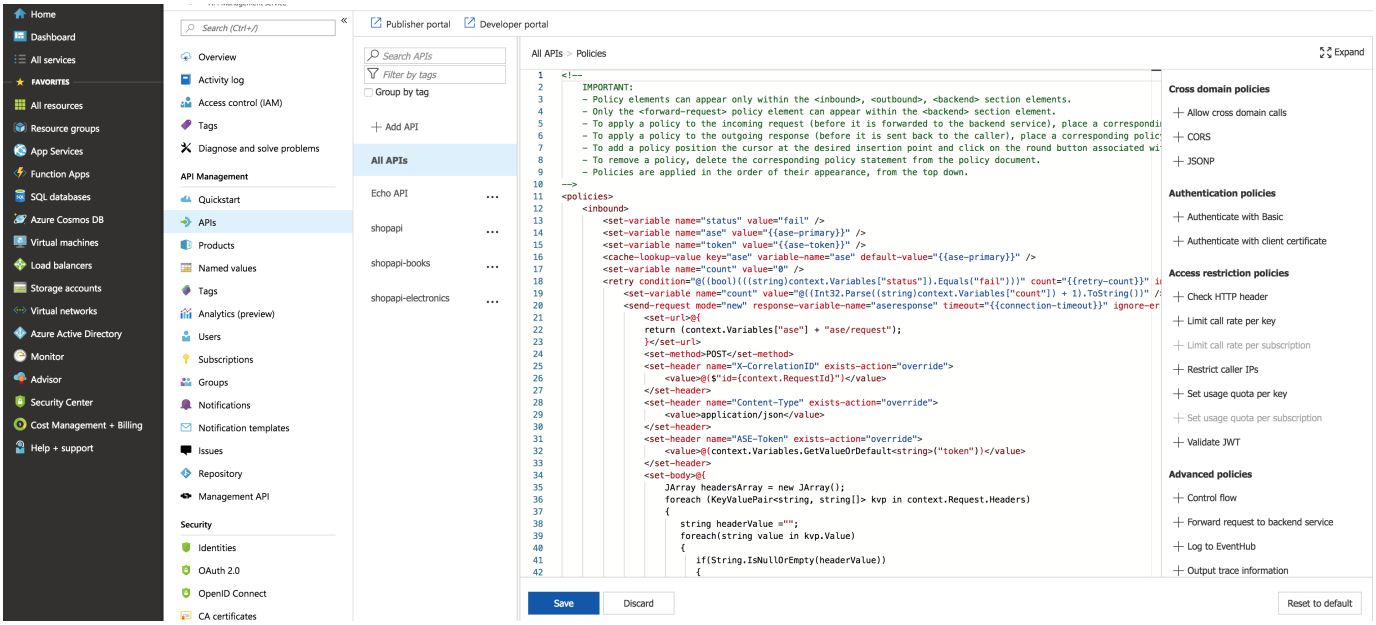
1. Download the Azure API Management policy XML file from the Sideband Integration section of the PingIntelligencedownloads page.
2. Open the downloaded Azure policy XML file and copy the policy at the desired level: all APIs, individual APIs, operation level, or group of APIs. Click Policies in the Inbound processing box and paste the policy.

> **Note**
>
> The PingIntelligence policy does not validate the authenticity of a JSON Web Token (JWT). Configure the PingIntelligence policy after the `<validate-jwt>` policy.



3. Click the Save button to save the policy.

Copyright © 2025 Ping Identity Corporation



> **Note**
>
> If an existing policy is deployed, copy and paste the `<inbound>` section of the PingIntelligence policy into the `<inbound>` section of your existing policy. Similarly, replace the `<outbound>` section of the policy. It is recommended that the PingIntelligence policy be the first policy that is executed.

Next steps

PingIntelligence API discovery is a process to discover and report APIs from your API environment. The discovered APIs are reported in PingIntelligence Dashboard. APIs are discovered when a global API JavaScript Object Notation (JSON) is defined in the ASE. For more information, see API discovery and configuration. You can edit the discovered API's JSON definition in the Dashboard before adding it to ASE. For more information on editing and configuring API discovery, see Discovered APIs.

Integrating PingIntelligence

After the policy deployment is complete, integrate PingIntelligence.

Before you begin

Refer to the ASE and ABS AI Engine Admin Guides.

About this task

To integrate PingIntelligence:

Steps

1. **Refer to the following:**
    - ASE ports
    - API naming guidelines
    - **Adding APIs in Sideband ASE.**

Copyright © 2025 Ping Identity Corporation

You can add individual APIs or you can configure a global API. For more information, seeAPI discovery and configuration.

- ASE to ABS connectivity

2. After you have added your APIs in ASE, train the API model. Refer to the following for guidance.

   The training of an API model is executed in the ABS artificial intelligence (AI) engine.

   - Managing AI engine training
   - API reports using Postman
   - Accessing the PingIntelligence Dashboard

Optional: Configuring ASE persistent connections

You can optionally configure TCP keep-alive connections in the `ase.conf` file of API Security Enforcer (ASE).

About this task

To configure TCP keep-alive connections:

Steps

- Set the `enable_sideband_keepalive` variable to `true`.

  The following snippet is from `ase.conf` displaying the default `enable_sideband_keepalive` variable. The default value is set to `false`.

```
; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false
```

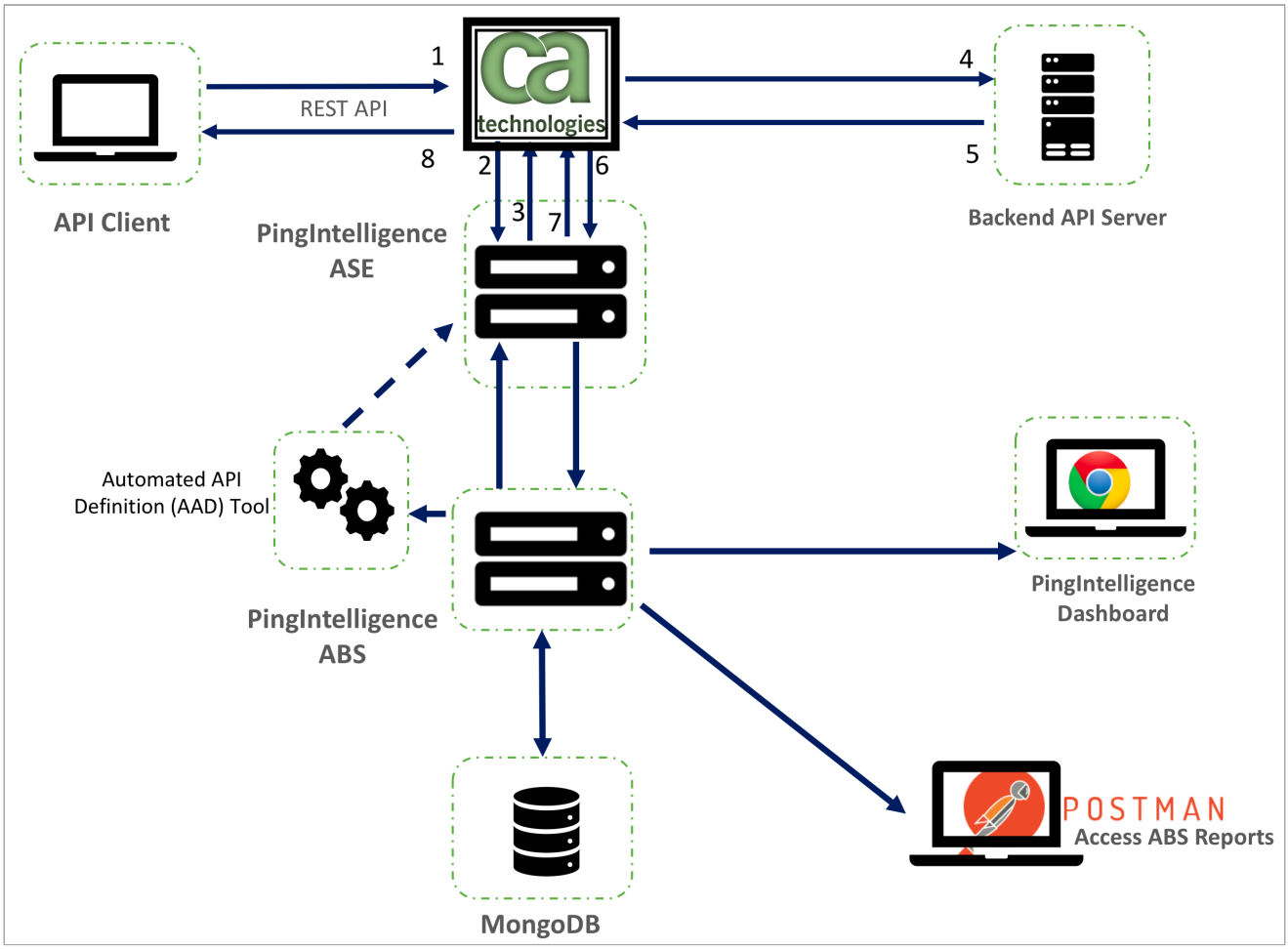
CA API gateway sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with CA API gateway.

You can attach the PingIntelligence for APIs integration to your APIs in the CA API Gateway by incorporating the Encapsulated Assertions to a subset of or to each API policies. When these Encapsulated Assertions are executed inside an API Gateway policy, the gateway passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.

The following diagram shows the logical setup of PingIntelligence for APIs and CA API gateway:

Copyright © 2025 Ping Identity Corporation



Here is the traffic flow through the CA API gateway and PingIntelligence for APIs components.

1. Incoming API Client request arrives at the CA API Gateway
2. A PingIntelligence assertion running on the CA API Gateway makes an API call to send the request metadata to PingIntelligence ASE
3. ASE checks the request against a registered set of APIs and looks for the origin IP, cookie, OAuth2 token, or API key in the PingIntelligence Blacklist. If all checks pass, ASE returns a `200-OK` response to CA. If the client is on the deny list and blocking is enabled, a `403 response` is sent to CA. The request information is also logged by ASE and sent to the AI engine for processing.
4. If CA receives a `200-OK` response from ASE, then it forwards the client request to the backend server. Otherwise, the CA blocks the client when a `403 response` is received.
5. The response from the backend server is received by CA.
6. CA makes a second API call to pass the response information to ASE.
7. ASE receives the response information and immediately sends a `200-OK` to CA. The response information is also logged by ASE and sent to the AI engine for processing.
8. CA sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

PingIntelligence encapsulated assertions include capabilities for enhanced sideband performance and availability including:

- Persistent SSL sessions: Support for flowing sideband calls across a persistent Secure Sockets Layer (SSL) session between the API Gateway and PingIntelligence.

  > **Note**
  >
  > Requires enabling `enable_sideband_keepalive` parameter in the PingIntelligence ASE `ase.conf` file.

- Redundant PingIntelligence nodes: Optional redundant PingIntelligence ASE nodes can be configured in the encapsulated assertion to bypass a node failure.

Preparing to deploy the CA API gateway integration

Confirm that the following prerequisites are met before deploying the PingIntelligence integration.

About this task

Before deploying the PingIntelligence integration:

Steps

1. Configure the gateway using CA API Gateway Policy Manager.

   PingIntelligence was developed with and qualified with CA API Gateway 9.4. Contact PingIdentity for other supported releases.

2. Install and configure the PingIntelligence 4.0 or higher software.

   For more information, refer to PingIntelligence manual deployment or PingIntelligence automated deployment for virtual machines and servers.

3. Install Java on the system from where the bundle is imported into the CA API Gateway.

4. Verify that ASE is operating in `sideband` mode by running the following command in the ASE command line:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

   ```
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : enabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
   ```

   Troubleshooting:

Copyright © 2025 Ping Identity Corporation

If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

5. For a secure communication between CA and ASE, enable sideband authentication by entering the following command in the ASE command line:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

6. To generate the token in ASE, enter the following command in the ASE command line. Save the generated authentication token for further use.

   A token is required for CA to authenticate with ASE. This token is generated in ASE and configured in the policy Extensible Markup Language (XML) file.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

Installing and configuring the PingIntelligence bundle

Install and configure the PingIntelligence bundle for the CA API Gateway.

About this task

To install and configure the PingIntelligence bundle:

Steps

1. Download the PingIntelligence policy files from the Ping Identity Download site.

   The downloaded package will have the following files and properties:

   - ASE Check Request: The assertion used to analyze API requests.
   - ASE Check Response: The assertion used to analyze API responses.
   - Cluster-wide Properties:
     - `ase_host_https`: The default is https://ase-server.example.com.
     - `ase_host2_https`: The default is https://ase-server-2.example.com.
     - `ase_path_request` and `ase_path2_request`: The default path is /ase/request.
     - `ase_path_response` and `ase_path2_response`: The default path is /ase/response.
   - API examples:
     - `/shop`: Example API that may be called by an external client. The API shows how to support both failing and non-failing policies.
     - `/shop/backend`: An example shop-backend for demo purposes.

2. Untar the package.

Copyright © 2025 Ping Identity Corporation

3. Edit the `pingintelligence-properties.bundle` to configure the following properties:

Property	Description
`ase_host_https` and `ase_host2_https`	Primary and secondary PingIntelligence API Security Enforcer (ASE) Internet Protocol (IP) address and port number. If the primary ASE is not available, the request is sent to the secondary ASE.
`ase_request_connection_timeout`	The time in milliseconds for which the API gateway waits to establish a TCP connection for the client request with ASE. After the timeout period, the request is directly sent to the backend server. The default value is 30,000 milliseconds.
`ase_request_read_timeout`	The time in milliseconds for which the API gateway waits to get a response from ASE for the request. After the timeout period, the request is directly sent to the backend server. The default value is 60,000 milliseconds.
`ase_response_connection_timeout`	The time in milliseconds for which the API gateway waits to establish a TCP connection with ASE for the response from the backend server. After the timeout period, the response is directly sent to the client. The default value is 30,000 milliseconds.
`ase_response_read_timeout`	The time in milliseconds for which the API gateway waits to get a response from ASE for the request. After the timeout period, the request is directly sent to the backend server. The default value is 60,000 milliseconds.
`ase_path_request` and `ase_path2_request`	Use the default value in the sample file.
`ase_path_response` and `ase_path2_response`	Use the default value in the sample file.

Example:

The following is a sample `pingintelligence-properties.bundle` file:

Copyright © 2025 Ping Identity Corporation

```
<?xml version="1.0" encoding="UTF-8"?><l7:Bundle xmlns:l7="http://ns.l7tech.com/2010/04/gateway-management">
            <l7:References>
                <l7:Item>
                    <l7:Name>ase_host_https</l7:Name>
                    <l7:Id>f33082fa66314439b5d7e8703ac0963a</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="f33082fa66314439b5d7e8703ac0963a" version="1">
                             <l7:Name>ase_host_https</l7:Name> <l7:Value>https://your-ase-host-and-port</
l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_path_request</l7:Name>
                    <l7:Id>f33082fa66314439b5d7e8703ac09636</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="f33082fa66314439b5d7e8703ac09636" version="0">
                             <l7:Name>ase_path_request</l7:Name> <l7:Value>/ase/request</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_path_response</l7:Name>
                    <l7:Id>f33082fa66314439b5d7e8703ac09633</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="f33082fa66314439b5d7e8703ac09633" version="0">
                             <l7:Name>ase_path_response</l7:Name> <l7:Value>/ase/response</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_request_connection_timeout</l7:Name>
                    <l7:Id>07b5ecd6fc3baca9518885b71dbcee8e</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="07b5ecd6fc3baca9518885b71dbcee8e" version="0">
                             <l7:Name>ase_request_connection_timeout</l7:Name> <l7:Value>30000</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_request_read_timeout</l7:Name>
                    <l7:Id>07b5ecd6fc3baca9518885b71dbcee90</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="07b5ecd6fc3baca9518885b71dbcee90" version="0">
                             <l7:Name>ase_request_read_timeout</l7:Name> <l7:Value>60000</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_response_connection_timeout</l7:Name>
```


Copyright © 2025 Ping Identity Corporation

```
                    <l7:Id>07b5ecd6fc3baca9518885b71dbcee92</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="07b5ecd6fc3baca9518885b71dbcee92" version="0">
                            <l7:Name>ase_response_connection_timeout</l7:Name> <l7:Value>30000</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_response_read_timeout</l7:Name>
                    <l7:Id>07b5ecd6fc3baca9518885b71dbcee94</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-09T20:18:03.316Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="07b5ecd6fc3baca9518885b71dbcee94" version="0">
                            <l7:Name>ase_response_read_timeout</l7:Name> <l7:Value>60000</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_path2_response</l7:Name>
                    <l7:Id>753f4df53a2f3daf040f9807a4f9a126</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-18T17:04:41.043Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="753f4df53a2f3daf040f9807a4f9a126" version="0">
                            <l7:Name>ase_path2_response</l7:Name> <l7:Value>/ase/response</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_path2_request</l7:Name>
                    <l7:Id>753f4df53a2f3daf040f9807a4f9a124</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-18T17:04:41.043Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="753f4df53a2f3daf040f9807a4f9a124" version="0">
                            <l7:Name>ase_path2_request</l7:Name> <l7:Value>/ase/request</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
                <l7:Item>
                    <l7:Name>ase_host2_https</l7:Name>
                    <l7:Id>753f4df53a2f3daf040f9807a4f9a122</l7:Id>
                    <l7:Type>CLUSTER_PROPERTY</l7:Type>
                    <l7:TimeStamp>2019-07-18T17:04:41.043Z</l7:TimeStamp>
                    <l7:Resource>
                        <l7:ClusterProperty id="753f4df53a2f3daf040f9807a4f9a122" version="1">
                            <l7:Name>ase_host2_https</l7:Name> <l7:Value>https://your-second-ase-host-and-
port</l7:Value>
                        </l7:ClusterProperty>
                    </l7:Resource>
                </l7:Item>
            </l7:References>
```

Copyright © 2025 Ping Identity Corporation

Importing the PingIntelligence policy

After configuring the PingIntelligence bundle, import it into the CA API Gateway.

About this task

PingIntelligence provides a script to import the policy. To import the bundle:

Steps

1. Open the `import_pingintelligence.sh` file in a text editor.
2. Configure the following values:

Value	Description
`GW`	The API gateway hostname and port
`GW_user admin:password`	The API gateway username
`GW_PASS_B64`	A Base64 encoded password used to encrypt and decrypt secure passwords

3. Run the `import_pingintelligence.sh` script.

   Result:

   After the import script is run, the PingIntelligence policy is installed in the API gateway.

4. To verify the policy import, connect to the API gateway using the CA API Gateway Policy Manager. Verify the PingIntelligence folder is visible in the lower left-hand side window.

   Example:

   The following is a sample `import_pingintelligence.sh` script:

Copyright © 2025 Ping Identity Corporation

```
!/usr/bin/env bash

# Configure the gateway host and port and user credentials
#
GW=localhost:8443
GW_USER=admin:password
GW_PASS_B64==

# Import the folder 'PingIntelligence'
#
curl -k -u $GW_USER -X PUT -H "Content-Type: application/xml" -H "L7-key-passphrase: $GW_PASS_B64"
"https://$GW/restman/1.0/bundle" -d @../docker-build/add-ons/ssg/policies/pingintelligence.bundle

# Import cluster properties that configure the PingIntelligence bundle
#
# ase_host_https
# ase_path_request
# ase_path_response
# ase_host2_https
# ase_path2_request
# ase_path2_response
# ase_request_connection_timeout
# ase_request_read_timeout
# ase_response_connection_timeout
# ase_response_read_timeout
#
curl -k -u $GW_USER -X PUT -H "Content-Type: application/xml" "https://$GW/restman/1.0/bundle" -d
@../docker-build/add-ons/ssg/policies/pingintelligence-properties.bundle
```

Configuring the certificate and ASE token

After the bundle is imported into the CA API Gateway, configure the certificate and ASE token using the CA API Policy Manager.

Before you begin

Ensure you have completed the steps in Preparing to deploy the CA API gateway integration. You will need the generated token from step 6 to configure the ASE token.

About this task

To configure the ASE token and certificate using CA API Policy Manager

Steps

1. Configure the certificate:
    1. In CA API Policy Manager, navigate to Tasks → Certificate, Keys and Secrets → Manage Certificates.
    2. Click Add and complete the steps to add a certificate.
    3. In the Specify Certificate Options step (step 3 in the UI wizard), select the Outbound SSL Connections check box and click Next.

Copyright © 2025 Ping Identity Corporation

4. In the Configure Validation step (step 4 in the UI wizard), select the Certificate is a Trust Anchor check box and click Finish.

2. Configure the ASE token:

    1. In the CA API Policy Manager, navigate to Tasks → Certificate, Keys and Secrets → Manage Stored Passwords.
    2. Select ase_token and click on properties.
    3. In the Stored Password Properties pop-up window, click Change Password.
    4. In the Enter Password pop-up window, enter the previously generated ASE token and click Ok.

To generate the token, refer to Preparing to deploy the CA API gateway integration.

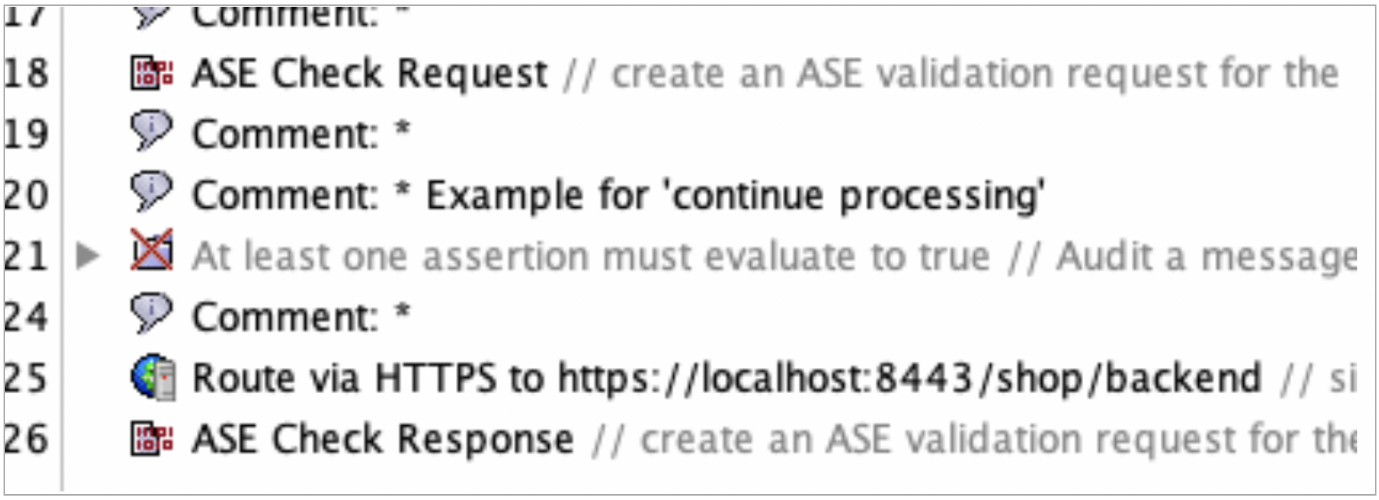
Apply the PingIntelligence policy

The PingIntelligence bundle includes API Security Enforcer (ASE) Check Request and Check Response encapsulated assertions.

Apply these assertions to each API that you want to monitor using PingIntelligence. You can include these assertions in global policies if you want each incoming API call to automatically be checked by PingIntelligence, or you can attach those assertions in service-level policies.

For service-level policies, each API will add two assertions:

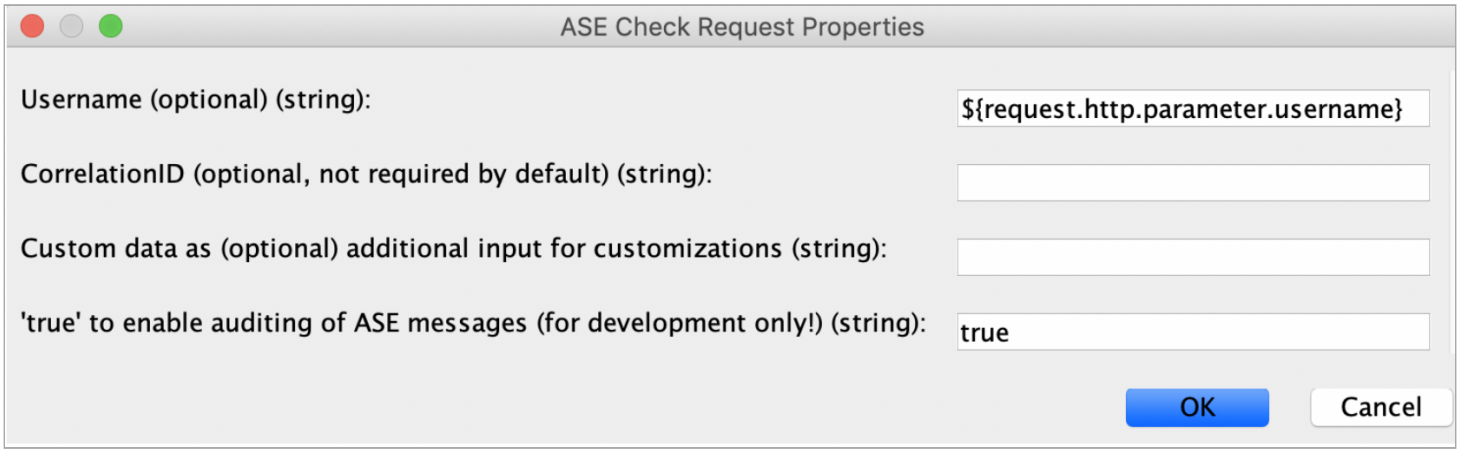
- ASE Check Request: Applied before routing the request to the backend
- ASE Check Response: Used after a call to the downstream endpoint (which is on line 25 in the image below)



ASE Check Request

The ASE Check Request assertion is configured with the following properties:

Copyright © 2025 Ping Identity Corporation



If you do not configure the properties, the assertion extracts all required details by itself. This includes:

- Retrieving all the request headers
- Generating a correlationId (used as X-CorrelationID)
- Retrieving the ASE token
- Retrieving the ASE HTTPS host
- Retrieving the ASE request path
- Sending a message to ASE

PingIntelligence recommends adding `username` to capture the user name when it is available. Examples of username variables include:

- `$\{request.http.parameter.username}`: The username variable included in the incoming request HTTP header
- `$\{session.subscriber_id}`: The username variable when authenticating users with the OAuth Toolkit (OTK)
- `$\{request.username}`: The username variable in the case of HTTP basic authentication

The variable name to use in this case will often be very implementation-specific. Use what you already defined as part of your CA API Gateway implementation.

You should change others if you are customizing to accommodate special use cases.

- `CorrelationID`: Optional, used if you want to override `correlationId`, which will otherwise automatically be assigned
- `Custom data`: Optional, used to modify the internal of that assertion
- `true`: Useful for users developing an API for debugging or auditing purposes

The assertion has an output that is the generated `correlationId:ase.correlationId` that is utilized by the ASE check response assertion.

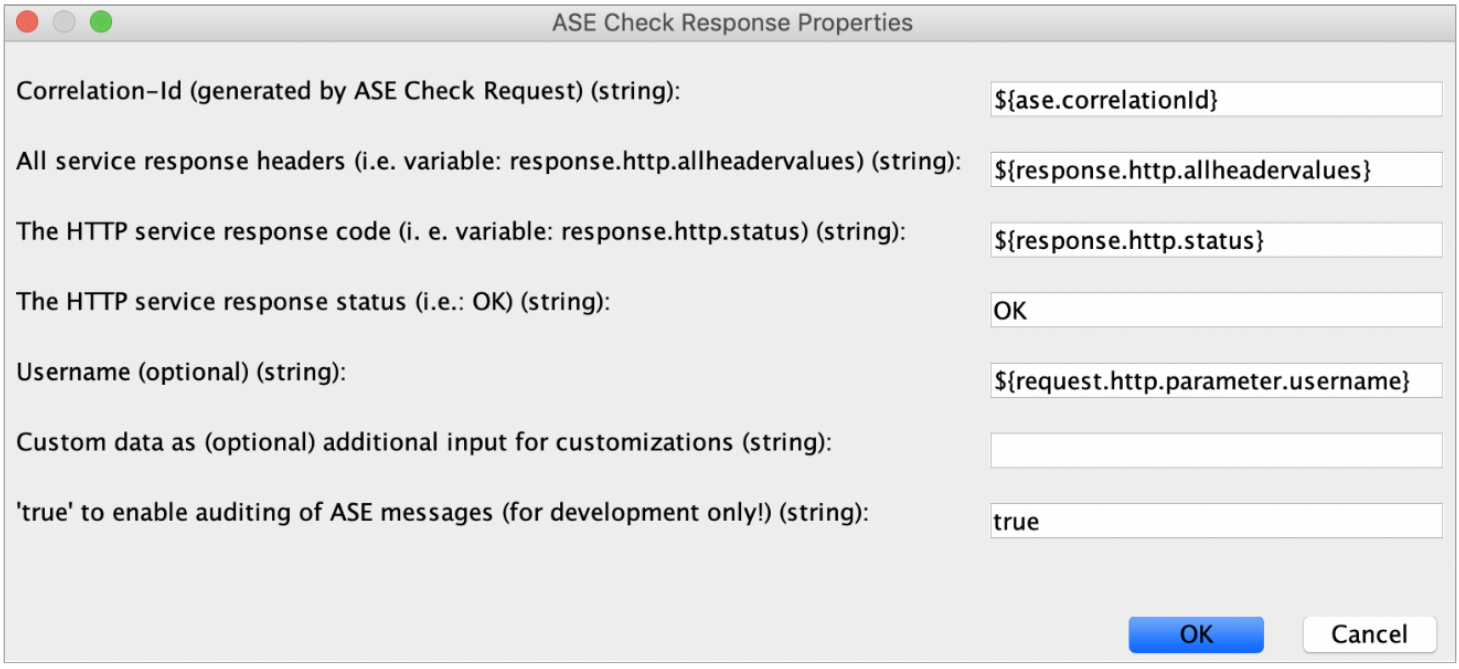
ASE Check Response

This ASE Check Response assertion must be configured for each API with the following variables:

Copyright © 2025 Ping Identity Corporation

.

Variable	Description
Correlation-ID	The ASE request and response correlation IDs, if specified, must match. Otherwise, keep `ase.correlationId`.
All service response headers	The default value is `$\{response.http.allheadervalues}`. This variable is created by the routing assertion that executed the backend call. If it is customized, for example, `myresponse`, then the updated variable should be used.
Response code	The HTTP response status of the backend call.
Response status	This value is ignored and hard coded to OK.
Username (optional)	This should match the username variable setting in the ASE Check Request assertion. The screenshot shows an example where the username is being extracted from the incoming HTTP request.
Custom data (optional)	Used by customers who would like to modify the internals of an assertion.
true	Useful for users developing an API for debugging or auditing purposes.



Copyright © 2025 Ping Identity Corporation

API discovery

PingIntelligence API discovery is a process to discover, and report APIs from your API environment. The discovered APIs are reported in the PingIntelligence Dashboard. APIs are discovered when a global API JavaScript Object Notation (JSON) is defined in the ASE.

For more information, see API discovery and configuration. You can edit the discovered API's JSON definition in the Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see Discovered APIs.

Integrating PingIntelligence

After the policy deployment is complete, integrate PingIntelligence.

Before you begin

Refer to the ASE and ABS AI Engine Admin Guides.

About this task

To integrate PingIntelligence:

Steps

1. Refer to the following:
    - ASE ports
    - API naming guidelines
    - Adding APIs in Sideband ASE.

    You can add individual APIs or you can configure a global API. For more information, seeAPI discovery and configuration.
    - ASE to ABS connectivity
2. After you have added your APIs in ASE, train the API model. Refer to the following for guidance.

    The training of an API model is executed in the ABS artificial intelligence (AI) engine.
    - Managing AI engine training
    - API reports using Postman
    - Accessing the PingIntelligence Dashboard

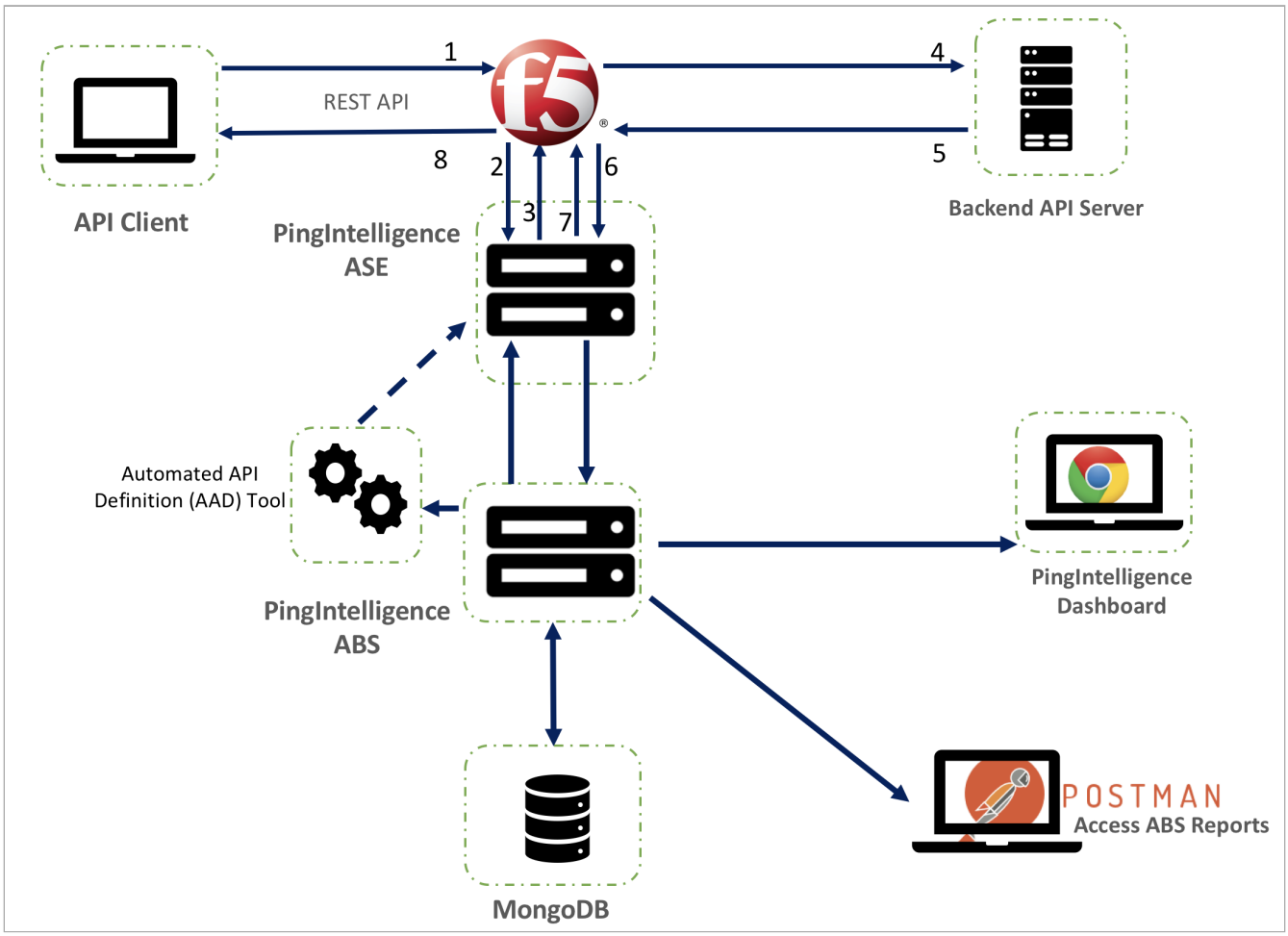
F5 BIG-IP integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with F5 BIG-IP.

A PingIntelligence policy is installed in F5 BIG-IP and passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking. PingIntelligence software includes support for reporting and attack detection based on usernames captured from JSON Web Token (JWT).

Copyright © 2025 Ping Identity Corporation

This diagram depicts the architecture of PingIntelligence for APIs components along with F5 BIG-IP:



The following is an description of the traffic flow through F5 BIG-IP and PingIntelligence ASE:

1. Client sends an incoming request to F5 BIG-IP.
2. F5 BIG-IP makes an API call to send the request metadata to ASE.
3. ASE checks the request against a registered set of APIs and looks for the origin IP, cookie, OAuth2 toke,n or API key in PingIntelligence artificial intelligence (AI) engine-generated deny list. If all checks pass, ASE returns a `200-OK` response to the F5 BIG-IP. If not, a different response code is sent to F5 BIG-IP. The request information is also logged by ASE and sent to the AI Engine for processing.
4. F5 BIG-IP receives a `200-OK` response from ASE, then it forwards the request to the backend server. A request is blocked only when ASE sends a `403 error` code.
5. The response from the backend server is received by F5 BIG-IP.
6. F5 BIG-IP makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to F5 BIG-IP.
8. F5 BIG-IP sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

Preparing to deploy the PingIntelligence policy

Before deploying the PingIntelligence policy, complete the following steps.

About this task

The F5 BIG-IP and PingIntelligence sideband integration was tested with F5 BIG-IP TMOS with Node.js v6.9.1. If you are using any other version of F5, contact Ping Identity support for help.

Before deploying the PingIntelligence policy:

Steps

1. Install and configure the following:
    1. F5 BIG-IP with v13.1.0.8 software.
    2. Knowledge of iRules LX in F5. Refer to the F5 documentation for information on iRules.
    3. A virtual server to front-end the incoming traffic. Make sure to applythe HTTP profile to the virtual server.
    4. A valid F5 BIG-IP license and iRules LX enabled in your setup.
2. Install and configure the PingIntelligence software.

   For more information, see PingIntelligence automated deployment for virtual machines and servers or PingIntelligence manual deployment.
3. Download the PingIntelligence policy from the Ping Identity Downloads site.
4. Sign on to your ASE machine and verify that ASE is in `sideband` mode by running the following `status` command:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

   ```
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : enabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
   ```

   Troubleshooting:

   If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.
5. For secure communication between F5 BIG-IP and ASE, enable sideband authentication by entering the following ASE command:

Copyright © 2025 Ping Identity Corporation

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p admin
```

6. To generate the token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use in Importing and configuring the PingIntelligence policy.

   A token is required for BIG-IP to authenticate with ASE.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

Importing and configuring the PingIntelligence policy

To deploy the PingIntelligence policy, first import and configure the policy.

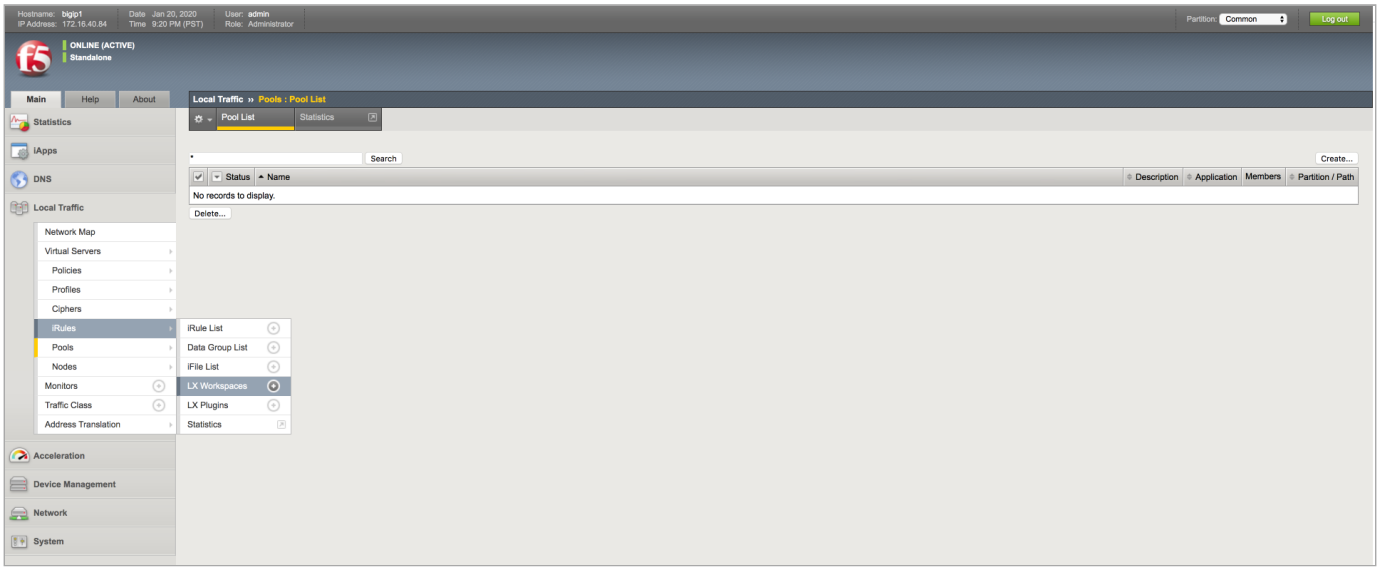
About this task

The PingIntelligence policy is specific to an ASE cluster. If you have more than one ASE cluster, then add the policy to a new workspace and create a new plugin. When you import the PingIntelligence policy, it is imported to an LX workspace and opens in a Node.js editor.

To import the PingIntelligence policy in F5:

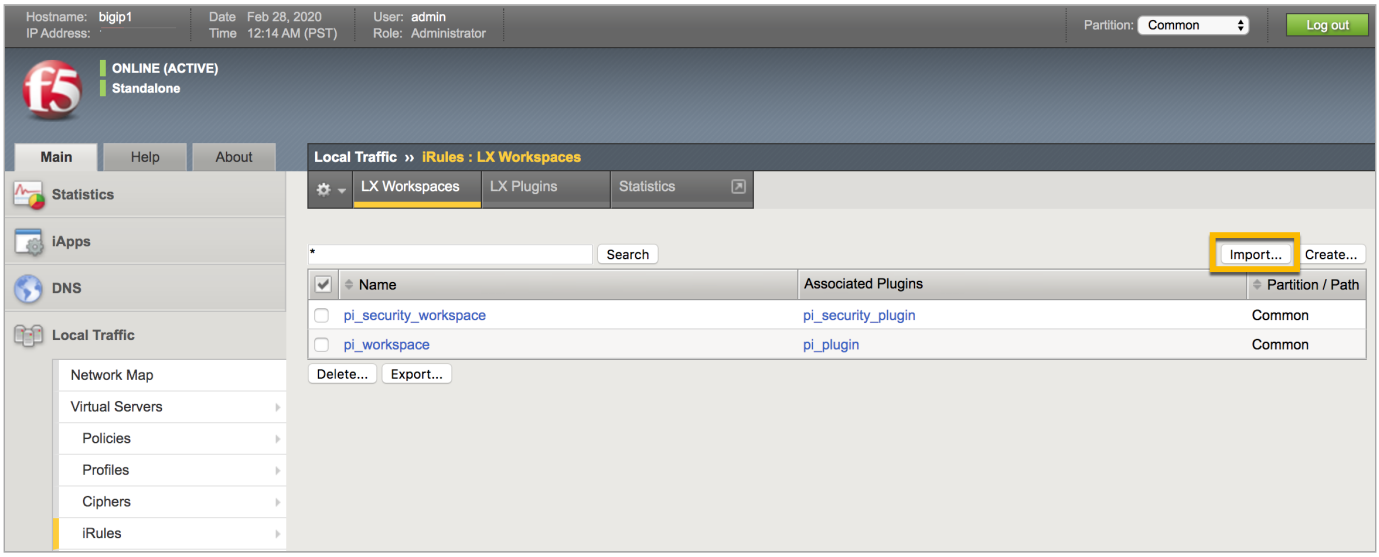
Steps

1. Sign on to F5 and navigate to Local Traffic → iRules → LX Workspaces.



2. In the Workspaces tab, click import.

Copyright © 2025 Ping Identity Corporation

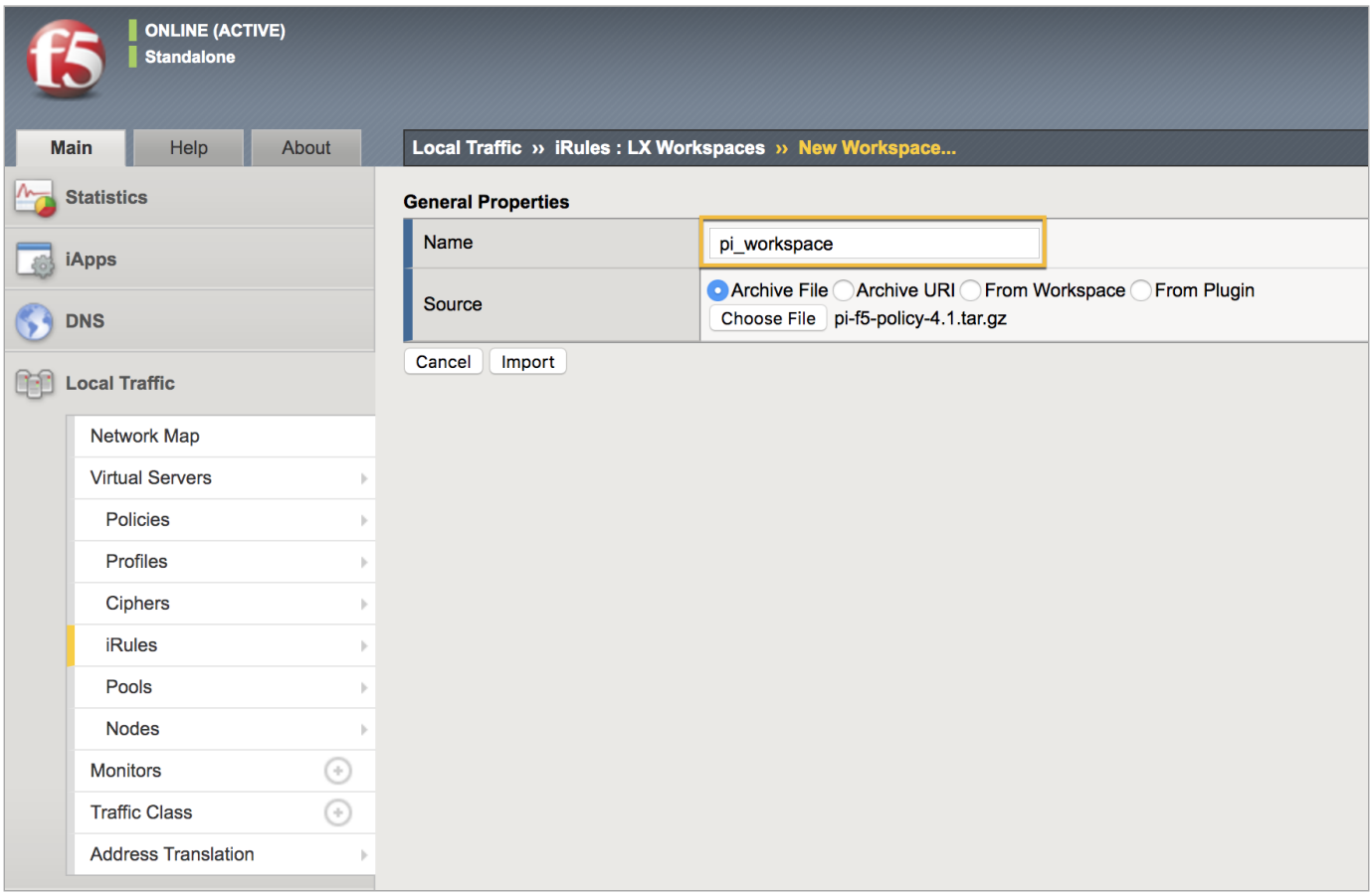


Result:

A Workspace import page is displayed.

3. Enter the Name and choose the PingIntelligence policy that you downloaded in Preparing to deploy the PingIntelligence policy.

4. Click Import.



Result:

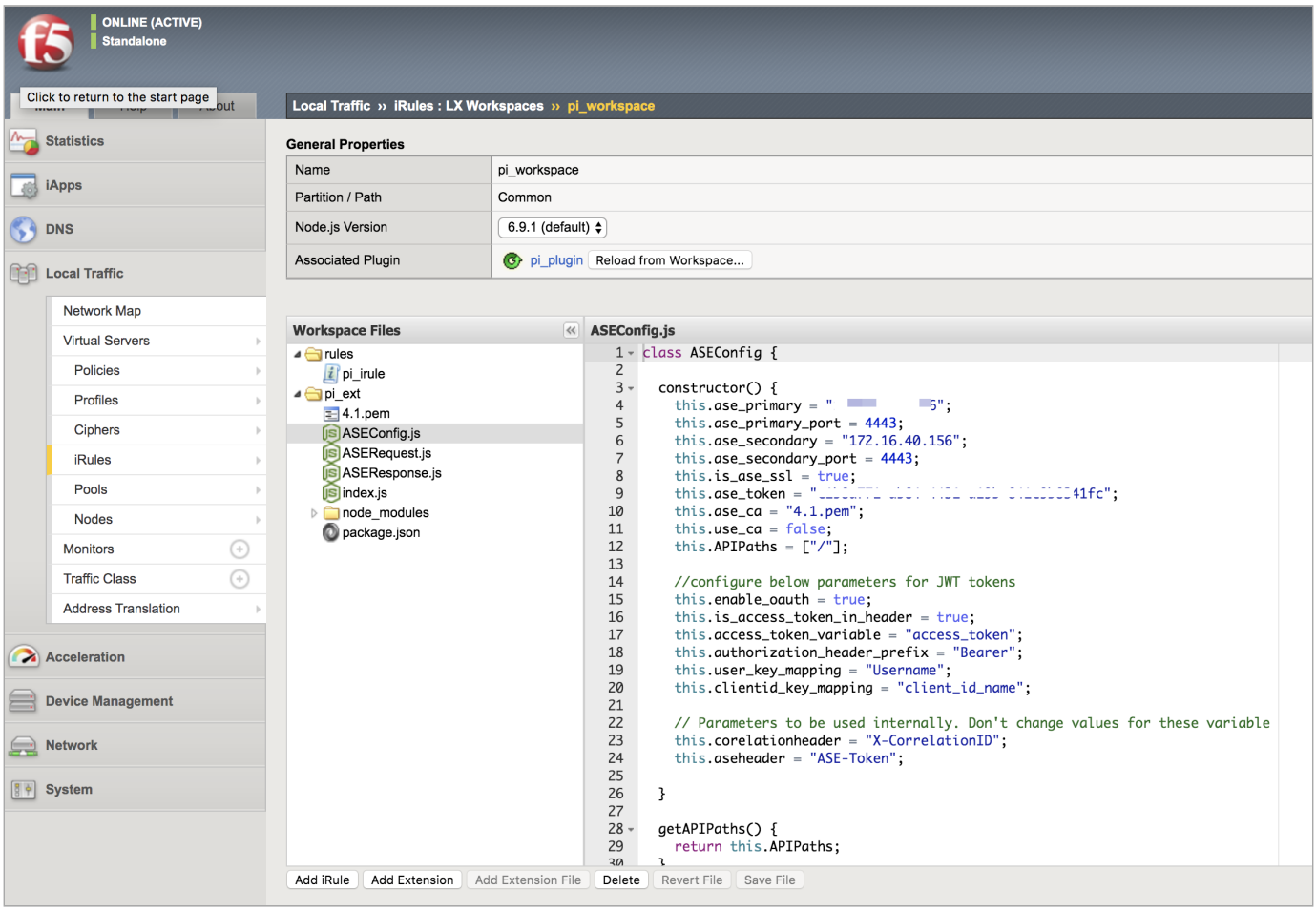
Copyright © 2025 Ping Identity Corporation

Clicking on Import creates an LX Workspace.

5. Click the Workspace.

   The policy is pre-loaded with the extension `oi_ext`.

6. Edit the ASE configuration by clicking the ASEConfig.js file.



The following table describes the ASE variables:

+

Variable	Description
`ase_primary`	IP address of primary ASE node
`ase_primary_port`	Port number of primary ASE node
`ase_secondary`	IP address of secondary ASE node
`ase_secondary_port`	Port number of secondary ASE node
`is_ase_ssl`	Set to `true` if traffic to ASE is sent over HTTPS

Copyright © 2025 Ping Identity Corporation

Variable	Description
`ase_token`	The ASE sideband authentication token that was generated as part of prerequisites
`use_ca`	Set to `true` if ASE is using a CA-signed certificate
`include_paths`	Provide the list of paths that the policy should process. If `/` is provided as path, then all the traffic is monitored. The maximum number of subpaths in path is 3. For example, `/a/b/c/`.
`enable_auth`	Set to `true` if traffic contains access token in `authorization` header or `querystring`.
`is_access_token_in_header`	Set to `true` if access token is present in `authorization` header.
`access_token_variable`	If the access token is present in `querystring`, then specify the key used for token.
`authorization_header_prefix`	If the access token is present in `authorization` header, then specify the prefix used for access token.
`user_key_mapping`	The location of `username` in JSON payload of JWT access token.
`clientid_key_mapping`	The location of client ID in JSON payload of JWT access token

+

Result:

+ The PingIntelligence policy opens in the editor.

Creating LX plugins

After importing and configuring the PingIntelligence policy, create an LX plugin named `pi_plugin`.

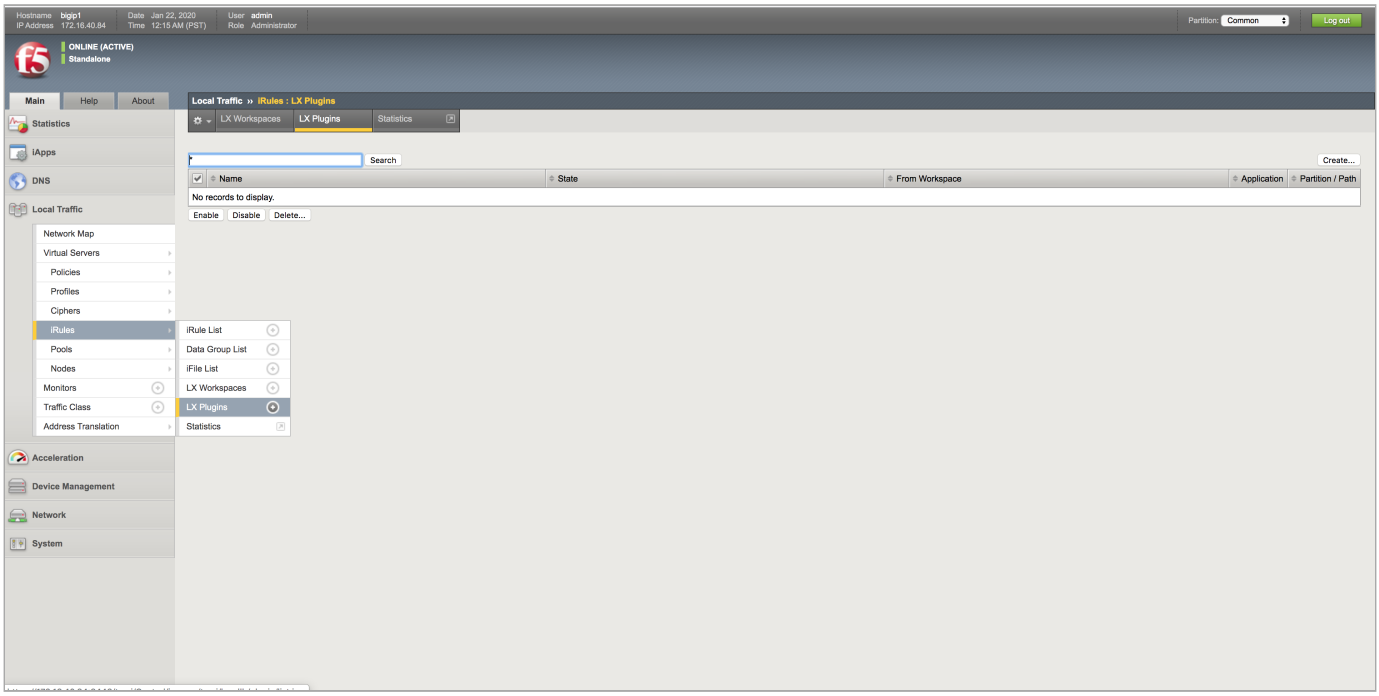
About this task

To create an LX plugin:

Steps

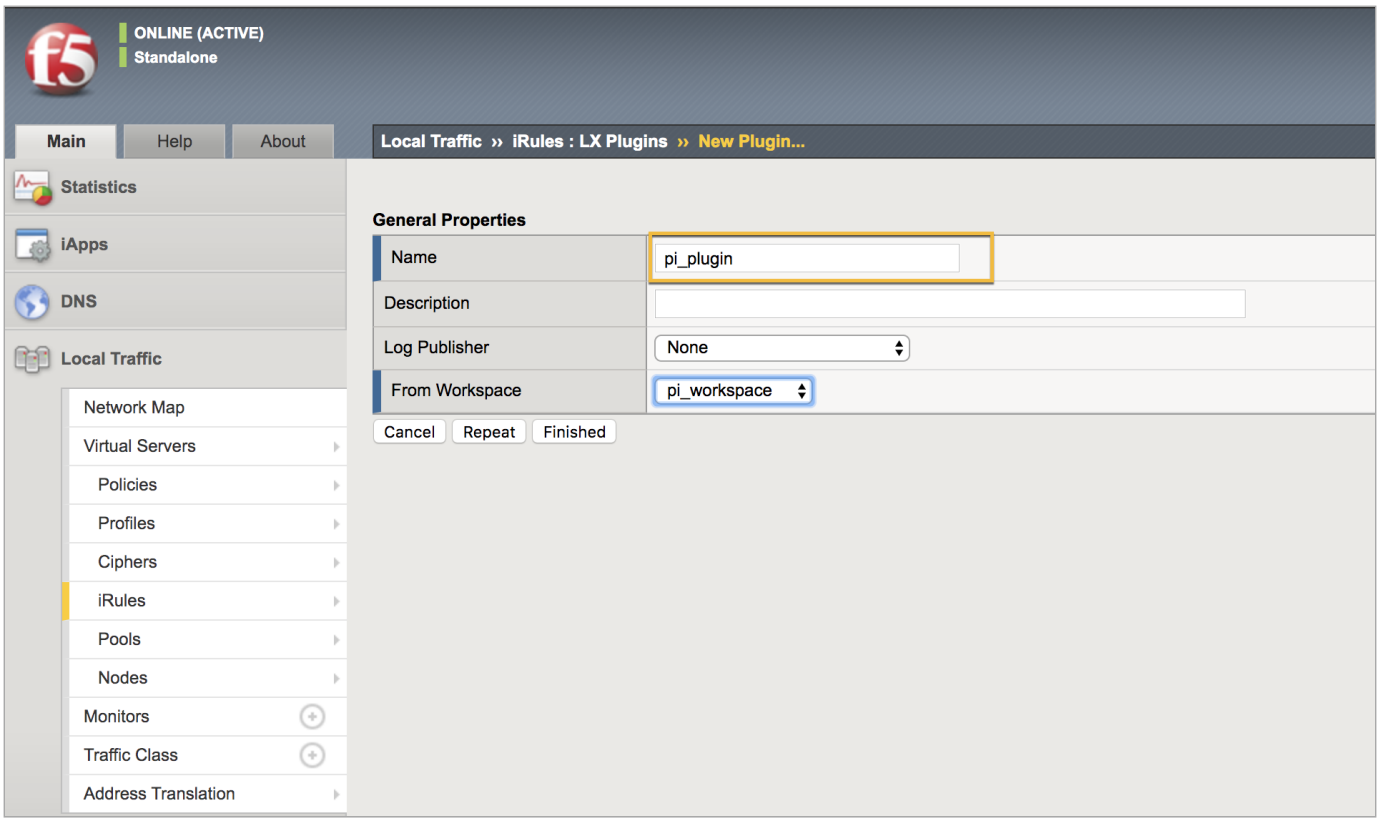
1. Navigate to Local Traffic → iRules → LX Plugins.

Copyright © 2025 Ping Identity Corporation



2. On the New Plugin page, click Create to create a new plugin named `pi_plugin`.

3. Select the workspace that you created in Importing and configuring the PingIntelligence policy from the From Workspace drop-down list and click Finished.



Copyright © 2025 Ping Identity Corporation

Optional: Creating a backend server pool and frontend virtual server

It is optional to create a backend server pool and frontend virtual server if you already have those set up.

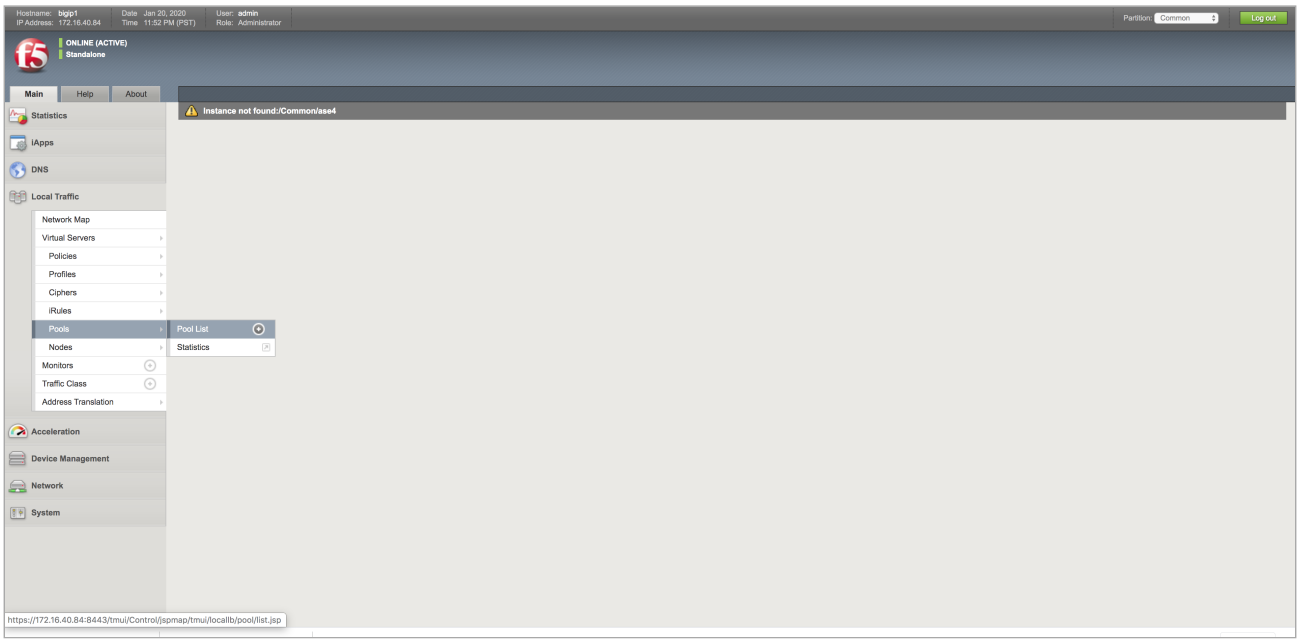
About this task

If you have an existing backend server pool and frontend virtual server that you want to use, skip to Adding the PingIntelligence policy.

If you do not have a backend server pool and frontend virtual server:

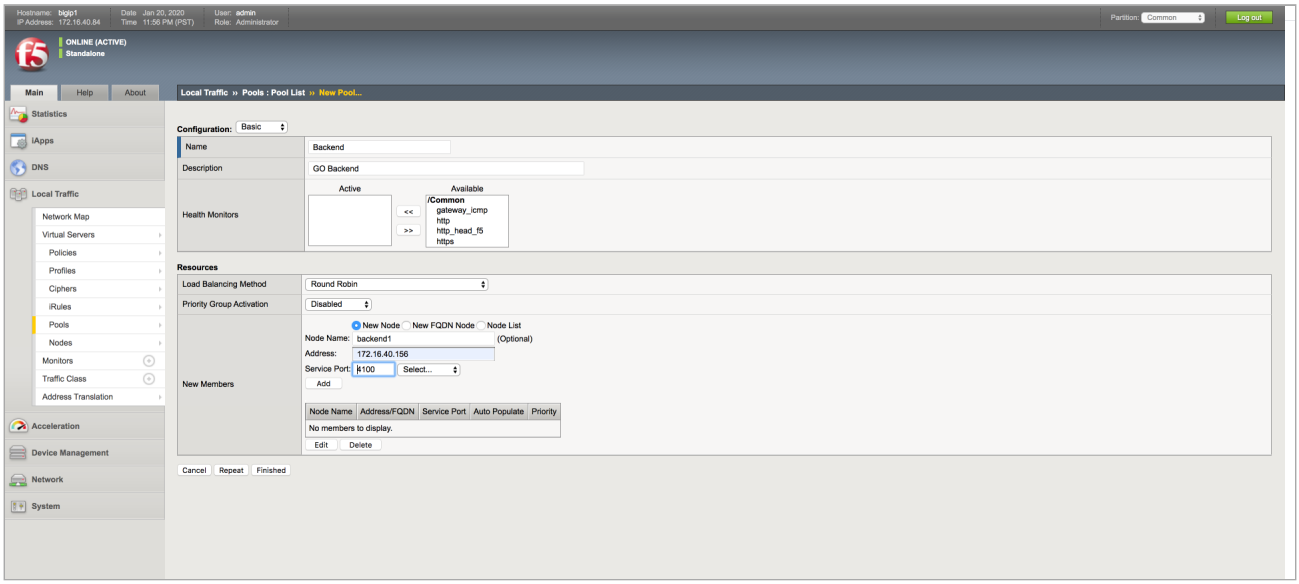
Steps

1. Create a backend server pool:
    1. Navigate to Local Traffic → Pools → Pool List and click Create.

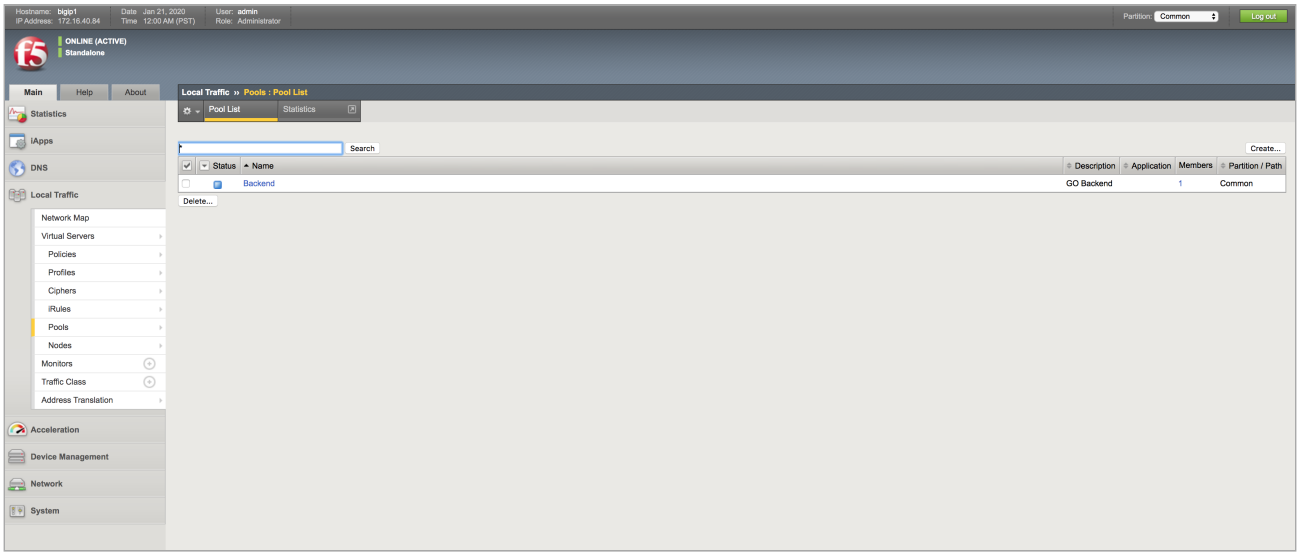


    2. In the Configuration page for the pool, configure the fields and add a new node for the backend.

Copyright © 2025 Ping Identity Corporation



3. Click Finished.



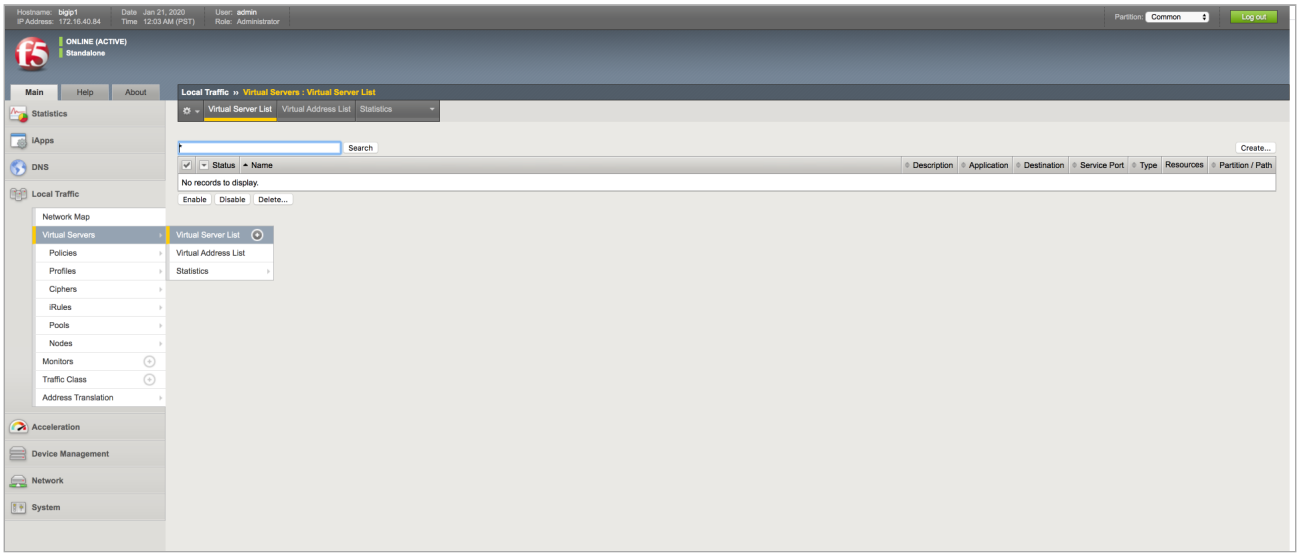
Result:

This creates a backend server pool, which is accessed from clients connecting to the frontend virtual server.

2. Add a frontend virtual server:

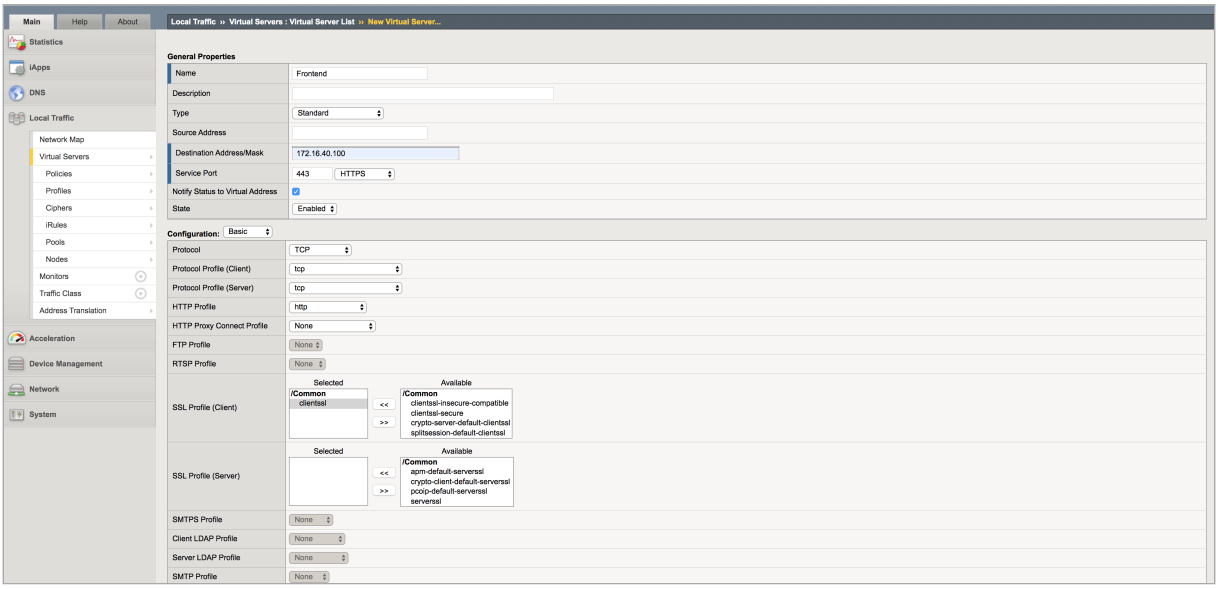
    1. Navigate to Local Traffic → Virtual Server → Virtual Server List and click Create.

Copyright © 2025 Ping Identity Corporation



2. Configure the frontend virtual server details. At a minimum, configure the following values:

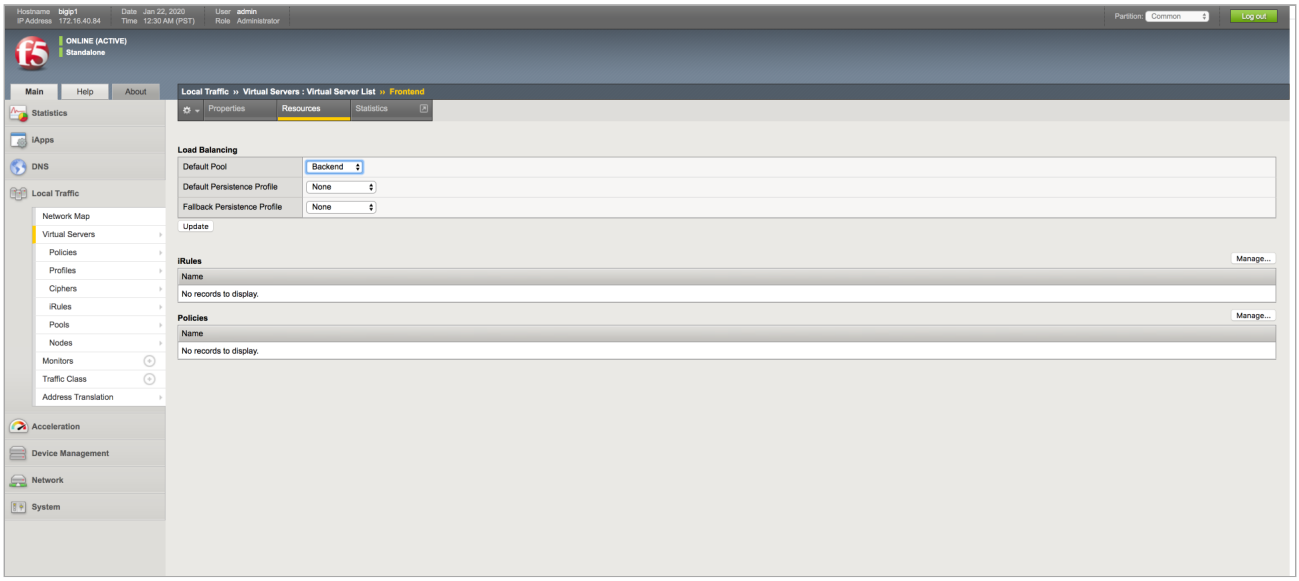
   - Destination Address: This is the virtual IP address that is used for the frontend.
   - SSL Profile (Client): Configure if the frontend is SSL.
   - SSL Profile (Server): Configure if the backend is SSL.



3. Click Finished.

4. Under the Resource tab, set the Default Pool as `Backend` and click Update.

Copyright © 2025 Ping Identity Corporation



Adding the PingIntelligence policy

The imported PingIntelligence policy must be tied to a virtual server. Add the PingIntelligence policy to the existing or recently created virtual server

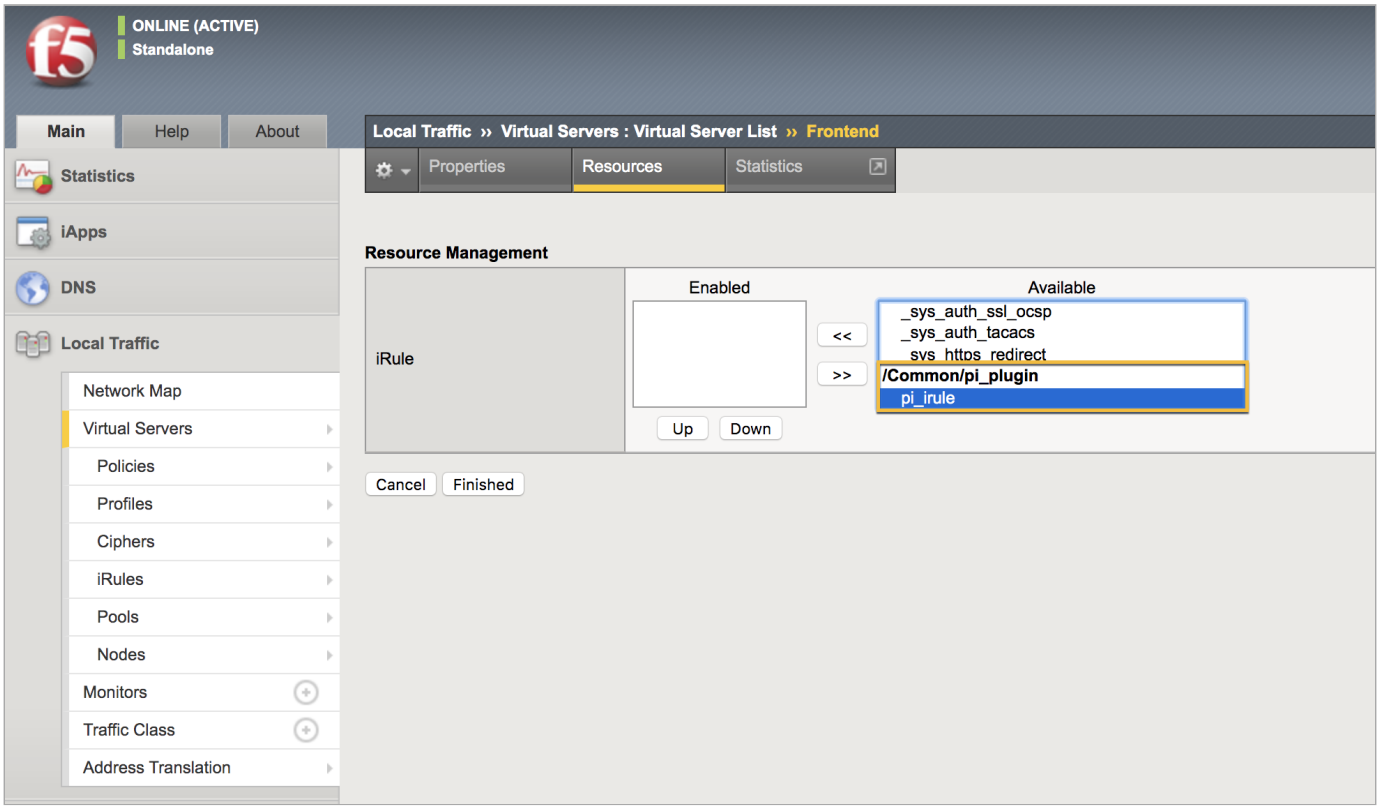
About this task

To add the PingIntelligence policy to the virtual server:

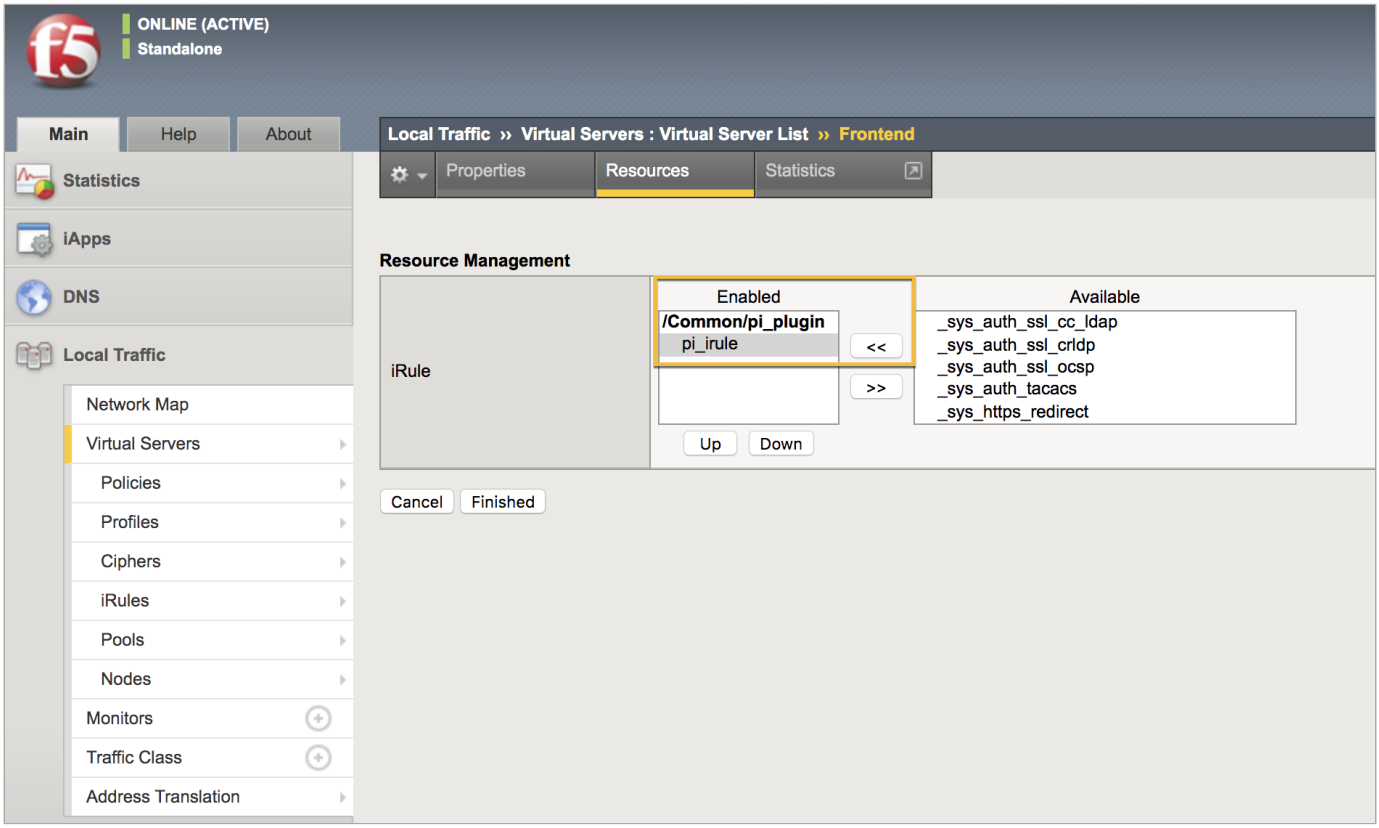
Steps

1. Navigate to Local Traffic → Virtual Servers → Virtual Server List.
2. Select the virtual server to which you want to add the PingIntelligence policy.
3. Click the Resources tab.
4. In the iRules section, click the Manage button.
5. Choose the iRule under the pi_plugin that you want to attach to the virtual server.

Copyright © 2025 Ping Identity Corporation



6. Move the pi_irule to the Enabled window and click Finished.



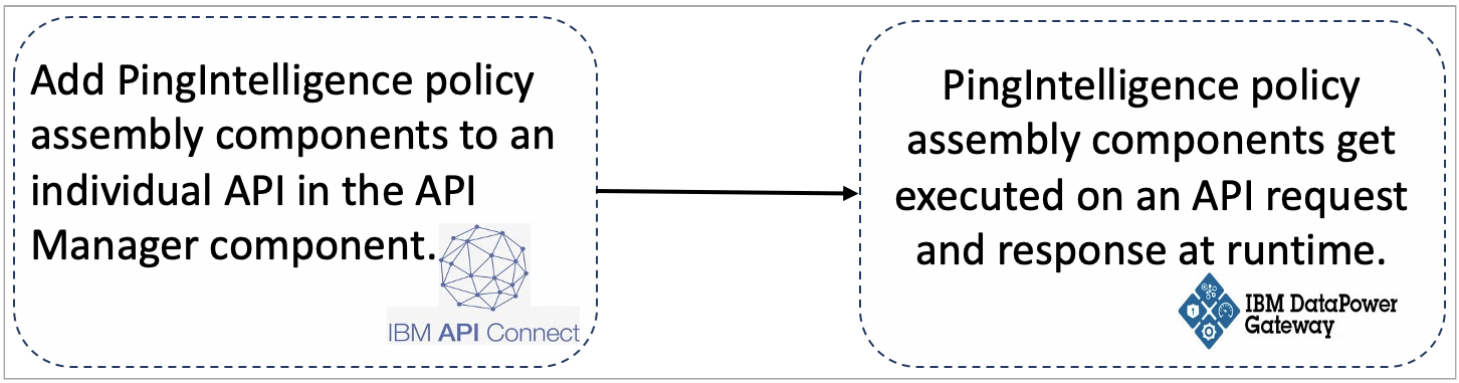
Copyright © 2025 Ping Identity Corporation

IBM DataPower Gateway sideband integration

This integration guide discusses the deployment of PingIntelligence for APIs in a sideband configuration with IBM DataPower Gateway.

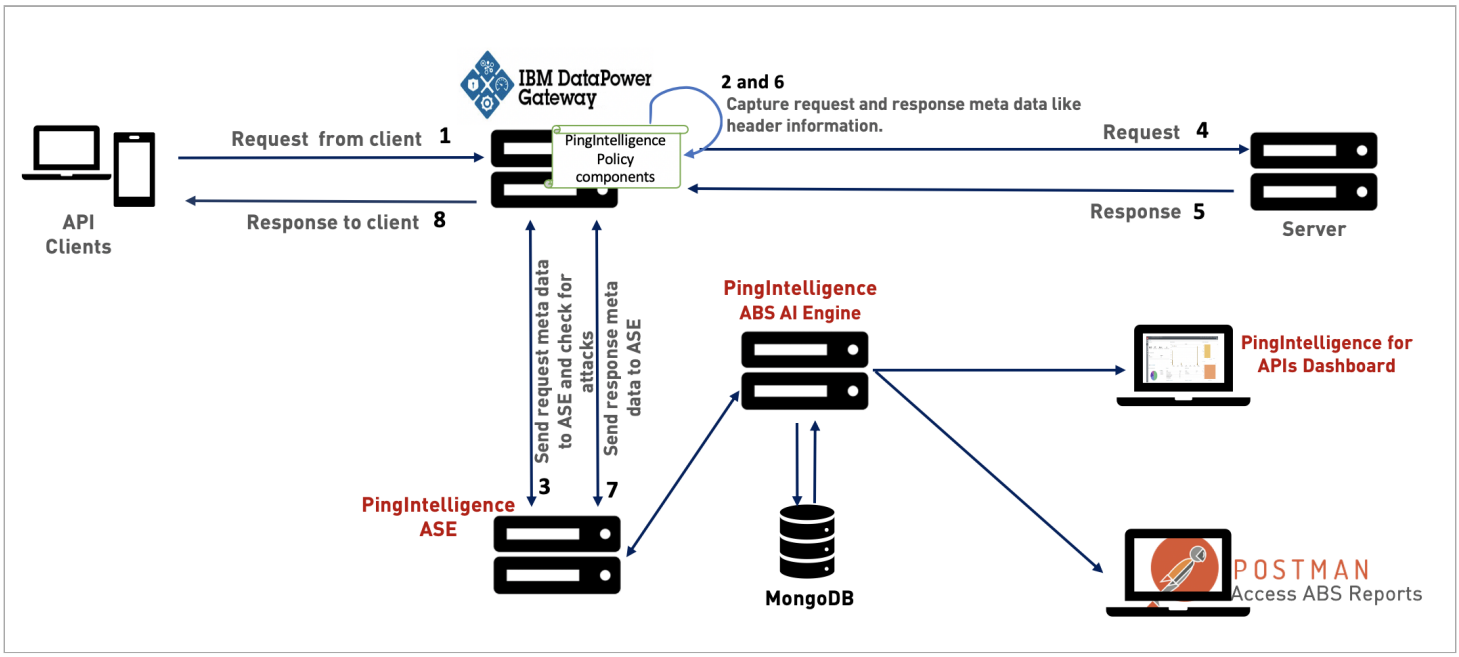
PingIntelligence for APIs provides policy assembly components that extract the API metadata from a request or response processed by IBM DataPower Gateway. The API metadata is passed to PingIntelligence for APIs for detailed API activity reporting and attack detection. For more information on sideband deployment, see Sideband ASE.

The PingIntelligence policy assembly components are added using API Manager in IBM API Connect. The following diagram shows the implementation steps of the PingIntelligence policy assembly components in the IBM API ecosystem.



> **Note**
>
> The PingIntelligence policy assembly components get deployed on a per API basis. You must configure them for an individual API to extract the request and response metadata for the API.

The following diagram shows the logical setup of PingIntelligence for APIs and IBM DataPower Gateway.



Copyright © 2025 Ping Identity Corporation

The traffic flow through the IBM DataPower Gateway and PingIntelligence for APIs components is explained below:

1. A client sends an incoming request to the IBM DataPower Gateway.
2. The PingIntelligence policy component is executed on the request to extract the metadata from the incoming request.
3. IBM DataPower Gateway makes an API call to send the request metadata to API Security Enforcer (ASE). The ASE checks the client identifiers such as usernames, tokens against the blacklist. If all checks pass, ASE returns a `200-OK` response to the IBM DataPower Gateway. If the checks do not pass, ASE sends different response code (403) to the IBM DataPower Gateway. In both cases, ASE logs the request information and sends it to the PingIntelligence API Behavioral Security (ABS) AI Engine for processing.
4. If the ASE sends a `200-OK` response to the IBM DataPower Gateway, it forwards the API requests to the backend server. If the gateway receives a `403-Forbidden` response from ASE, it blocks the client.
5. IBM DataPower Gateway receives the response from the backend server.
6. The PingIntelligence policy component is applied on the response to extract the metadata from the server response.
7. IBM DataPower Gateway makes a second API call to pass the response information to ASE, which sends the information to the ABS AI engine for processing.
8. IBM DataPower API Gateway sends the response received from the backend server to the client.

Preparing to deploy the PingIntelligence policy

Prepare to deploy the PingIntelligence policy by completing the following.

About this task

Before deploying the PingIntelligence policy:

Steps

1. Verify that the following versions of IBM APIC and DataPower are installed.

   The PingIntelligence policy is validated only for these versions

   - IBM APIC v5.0.8.7
   - IBM DataPower Gateway 2018.4.10
2. To configure the PingIntelligence policy, verify you have permissions to edit and publish APIs in the API Manager.
3. Install and configure the PingIntelligence software.

   For more information on PingIntelligence deployment, see PingIntelligence automated deployment for virtual machines and servers and PingIntelligence manual deployment.
4. Verify that API Security Enforcer (ASE) is in sideband mode by running the following ASE command:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

Copyright © 2025 Ping Identity Corporation

```
API Security Enforcer
status                  : started
mode                    : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : enabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

Troubleshooting:

If ASE is not in sideband mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set mode as `sideband` and start ASE.

For more information on starting ASE, see Starting and stopping ASE.

5. For a secure communication between IBM DataPower Gateway and ASE, enable sideband authentication by entering the following ASE command:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

6. Ensure SSL is configured in ASE for client side connection using self-signed certificate.

   For more information on configuring self-signed certificate, see Configuring SSL for external APIs.

7. Enable a connection keep-alive between gateway and ASE:

   1. Optional: If the ASE is running, stop it.
   2. Navigate to `/opt/pingidentity/ase/config/`.
   3. Set the value of `enable_sideband_keepalive` to `true` in the `ase.conf` file.
   4. Start ASE after setting the value.

   For more information on ASE configuration, see Sideband ASE configuration using the `ase.conf` file.

8. To generate the token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

   The token is required for IBM DataPower Gateway to authenticate with ASE. It is set as a runtime variable in ASE config set-variable policy. For more information, see Configuring the PingIntelligence policy components.

Copyright © 2025 Ping Identity Corporation

Deploy the PingIntelligence policy

PingIntelligence for APIs provides a `pi_policy.yaml` file for IBM DataPower Gateway sideband integration.

The policy has the following three policy assembly components:

- ASE Config: This assembly component configures the ASE connection and authentication parameters. It implements a set-variable policy that configures the parameters as runtime variables.
- ASE Request: This assembly component extracts the API metadata from a request processed by the IBM DataPower Gateway. It implements a gateway script policy in the DataPower Gateway.
- ASE Response: This assembly component extracts the API metadata from a response processed by the IBM DataPower Gateway. It implements a gateway script policy in the DataPower Gateway.

IBM API Connect provides different policy types to control specific aspects of processing by the DataPower Gateway (for example, to configure a capability for logging, for security, and so forth). The `set-variable` policy type helps to add or set a runtime variable. The `gateway` script policy gives built-in access to the DataPower Gateway to execute a specified DataPower Gateway script program.

The deployment of the PingIntelligence policy involves:

1. Adding the PingIntelligence policy components.
2. Configuring the PingIntelligence policy components.

> **Note**
>
> The PingIntelligence policy does not support payload with a DELETE request. When the policy is deployed, if a DELETE request comes with a payload, the payload will not reach the backend API server.

Adding the PingIntelligence policy components

Add the PingIntelligence policy components to your API.

Before you begin

Make sure to:

1. Download the PingIntelligence policy from the Ping Identity Downloads site.
2. Extract the policy by using the following command:

   ```
   # tar -zxvf  <<file name>>
   ```

   For example:

   ```
   # tar -zxvf pi-api-ibm-policy-4.1.0.tar.gz
   ```

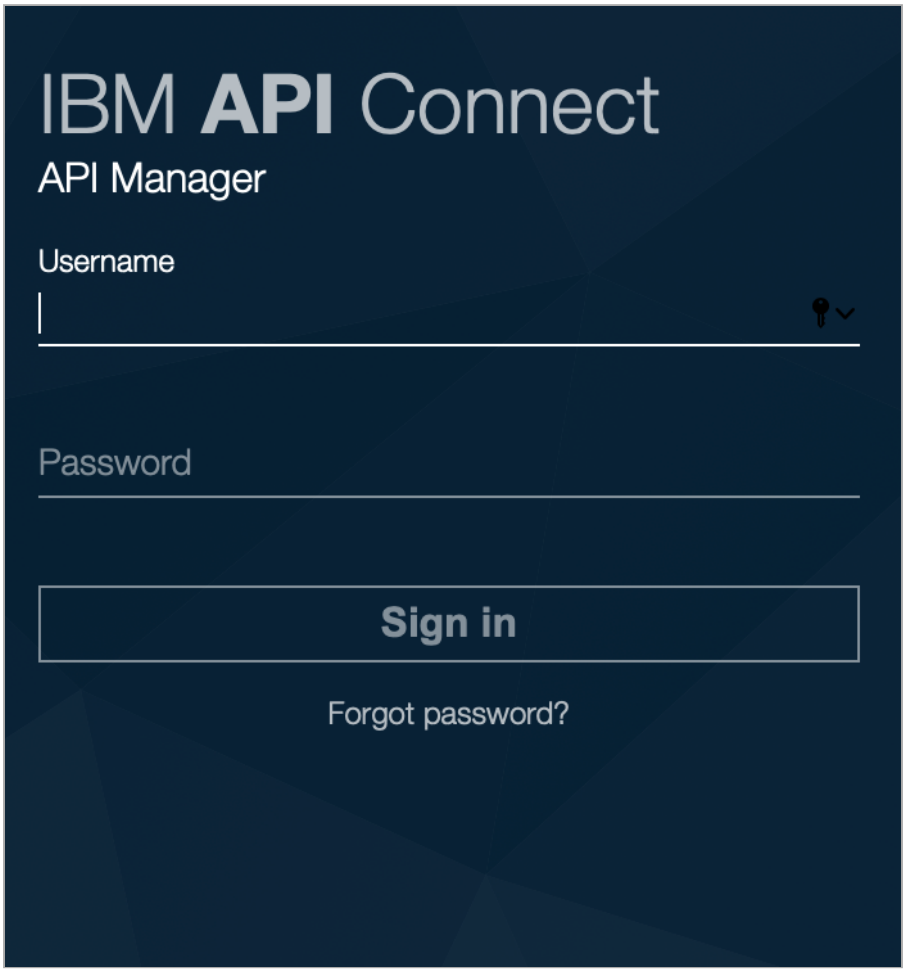
About this task

Copyright © 2025 Ping Identity Corporation

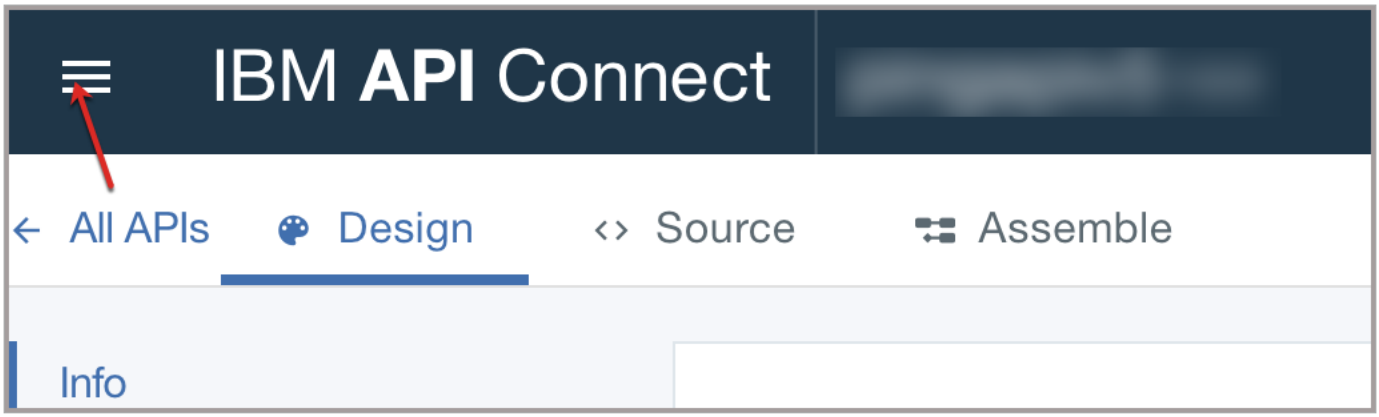
To add the PingIntelligence policy components:

Steps

1. **Sign on to IBM API Manger.**

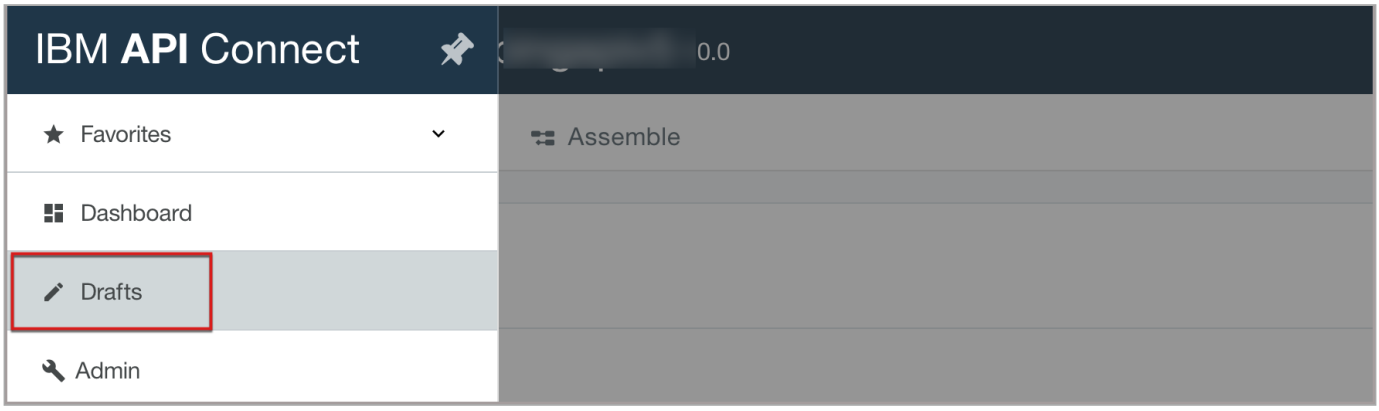


2. **To open the navigation pane, click the Menu**  **icon on the top-left corner.**

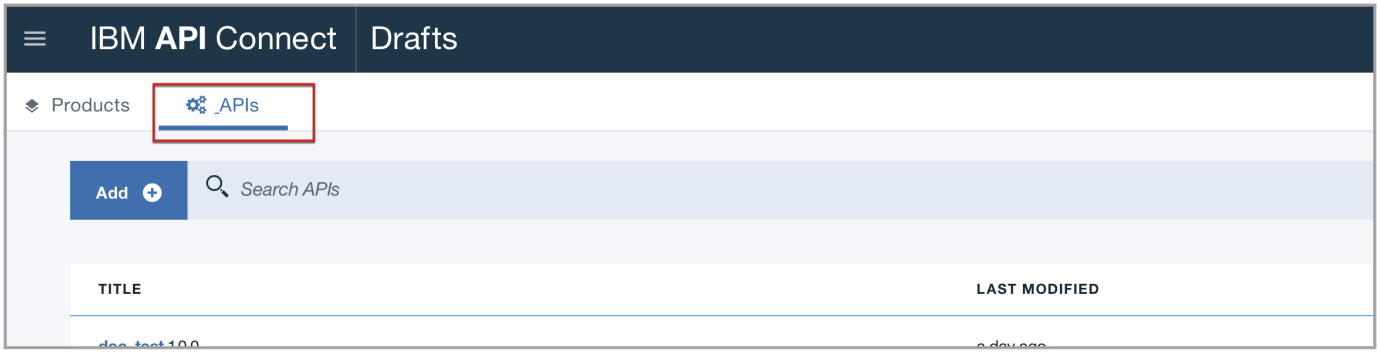


3. **Click Drafts in the navigation pane.**

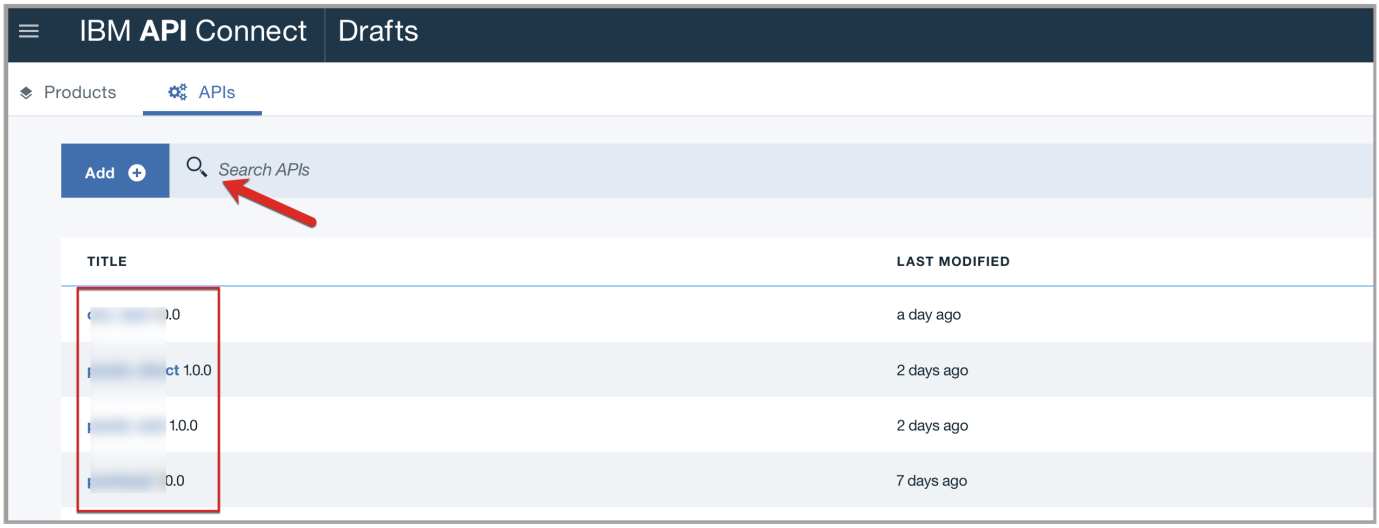
Copyright © 2025 Ping Identity Corporation



4. Click the APIs tab.

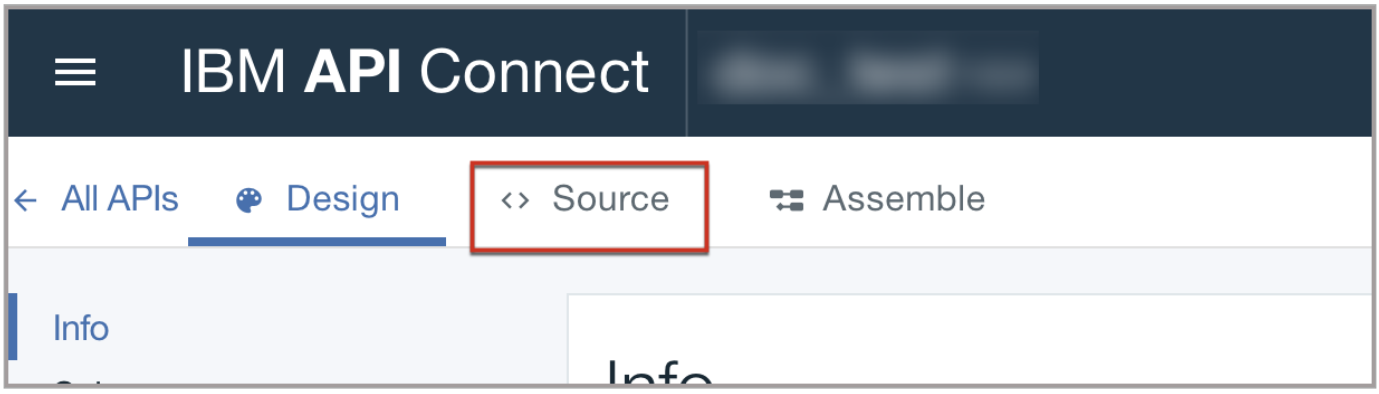


5. Click on your API under TITLE list or enter the API name in the Search APIs dialog box and select the API.



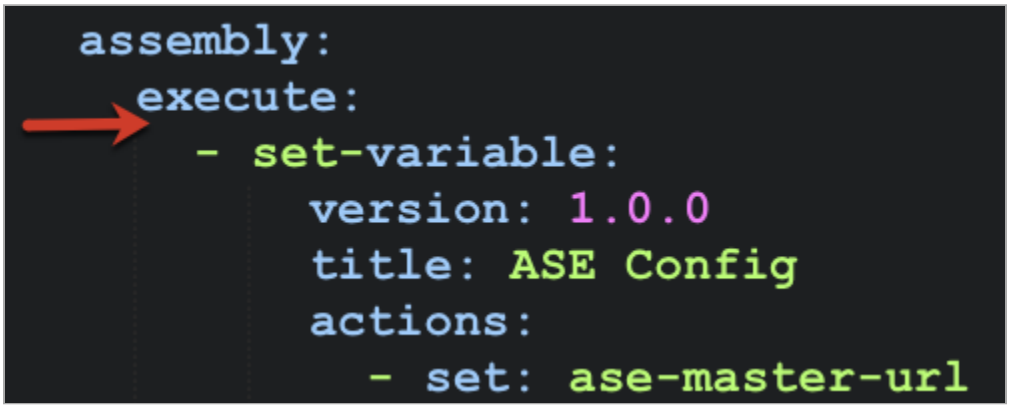
6. Click the Source tab to edit your API definition.

Copyright © 2025 Ping Identity Corporation

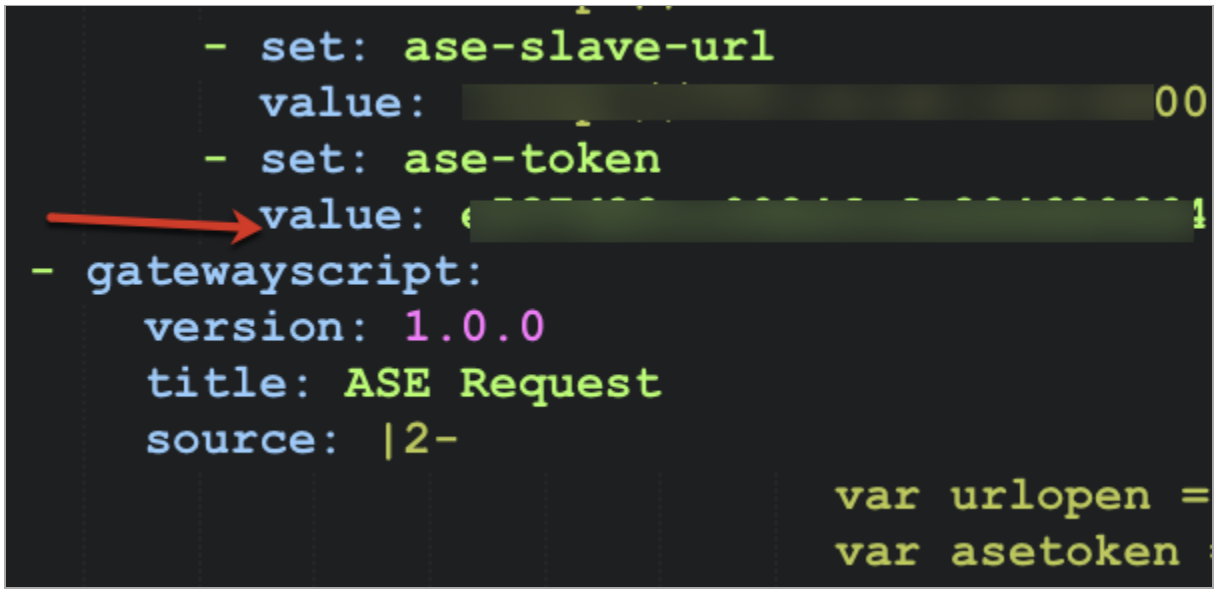


7. Copy and paste the contents of the PingIntelligence policy into the Assembly block of your API definition at three places:

    1. Open the `pi_policy.yaml` file, copy the contents of the set-variable: block with the ASE Config component and paste it in the next line after the execute: block in your API.



    2. Next, copy the contents of the gateway script: block containing the ASE Request component from the `pi_policy.yaml` file and paste it after the last line of the ASE Config component.



Copyright © 2025 Ping Identity Corporation

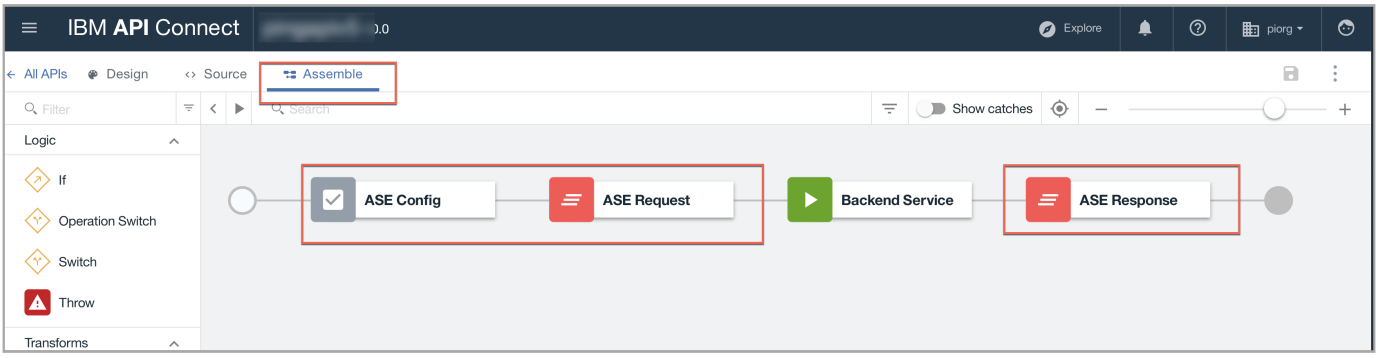
3. Copy the contents of the gateway script: block containing the ASE Response component from the `pi_policy.yaml` file and paste it as the last component of your API.

4. Copy the contents of the gateway script: block containing the ASE Response component from the `pi_policy.yaml` file and paste it as the last component of your API.

> **Note**
>
> The assembly component ASE Reponse should always be the last component of your policy assembly.

8. Click the Validate icon to validate your changes, and then click the Save icon after completing the validation.
9. Click the Assemble tab to open the Assemble view. Verify the sequence of the components ASE Config, ASE Request, and ASE Response in the Policy Assembly.

   The order must match as highlighted in the red boxes in the following image.



Configuring the PingIntelligence policy components

After adding the PingIntelligence policy to an API, configure the ASE parameters.

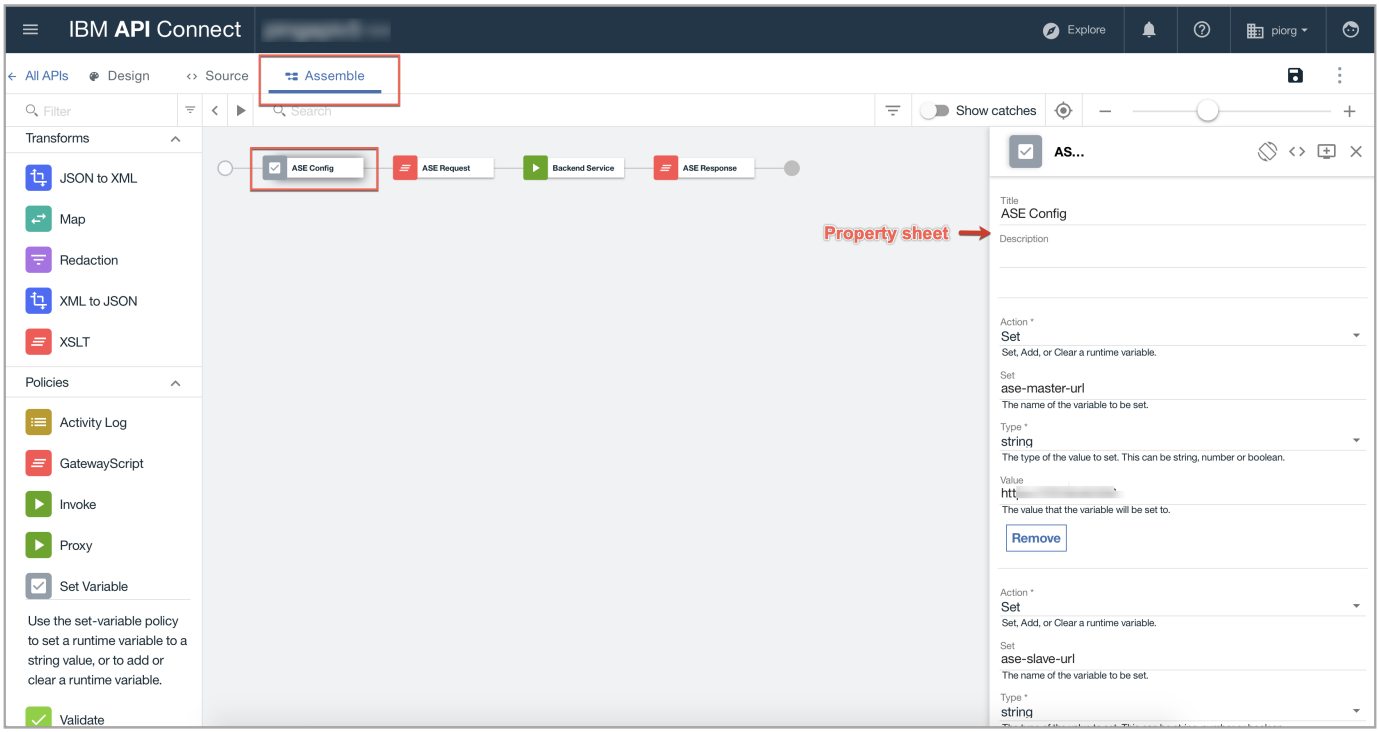
Copyright © 2025 Ping Identity Corporation

About this task

To configure the ASE parameters:

Steps

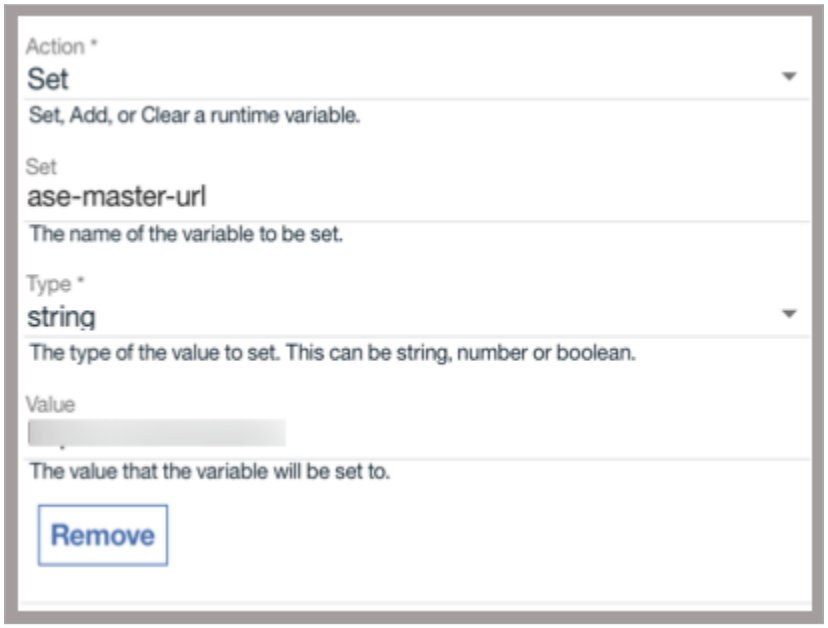
1. **Click the Assemble tab.**
2. **In the main window, click the ASE Config component to open the property sheet on the right.**

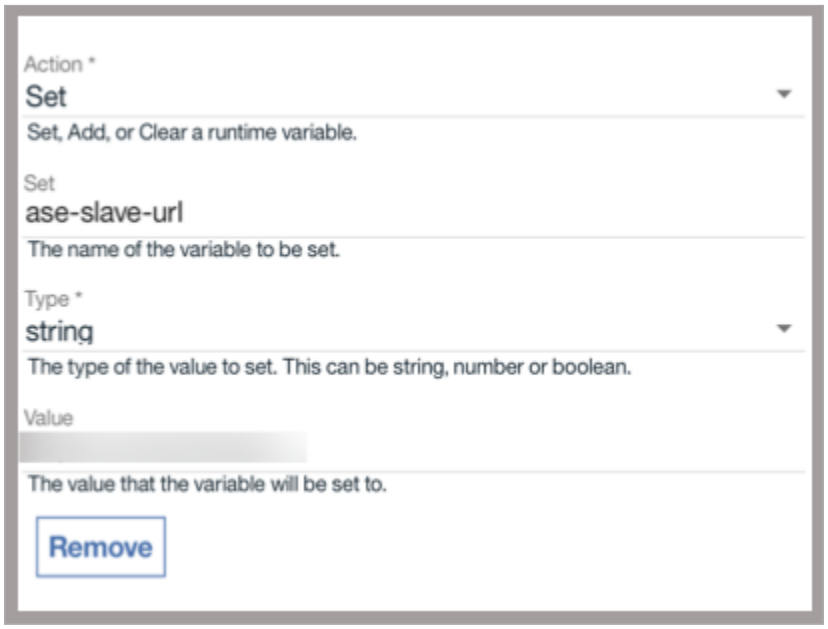


3. **Configure the values for ASE master URL, ASE slave URL, and ASE token. Click the Save icon on the top-right corner.**

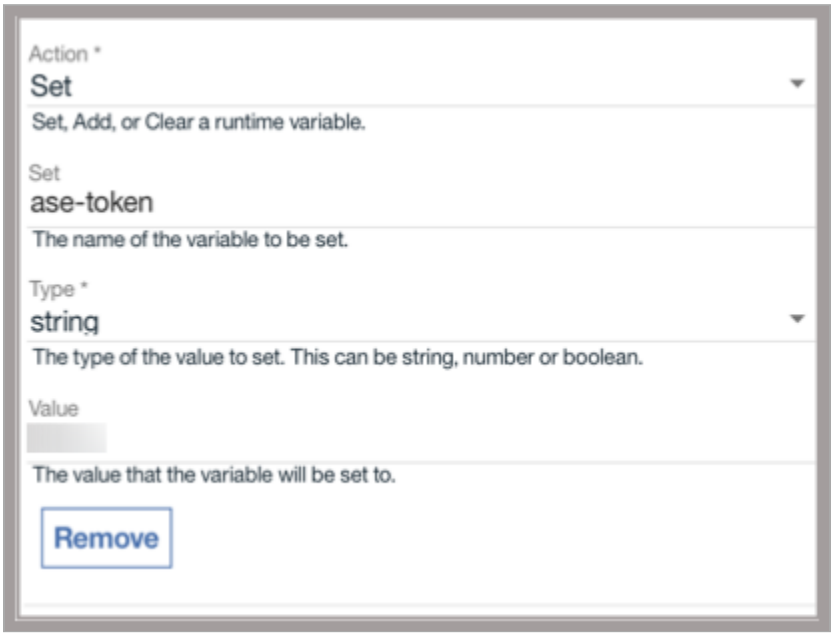
> **Note**
>
> The following format is applicable for ASE master and slave URLs: *<http/https>://<ASE-Host name or IP address>*.

Copyright © 2025 Ping Identity Corporation





Copyright © 2025 Ping Identity Corporation



4. Publish your API after completing step 3 to make the PingIntelligence policy components part of your API definition.

Kong API gateway integration

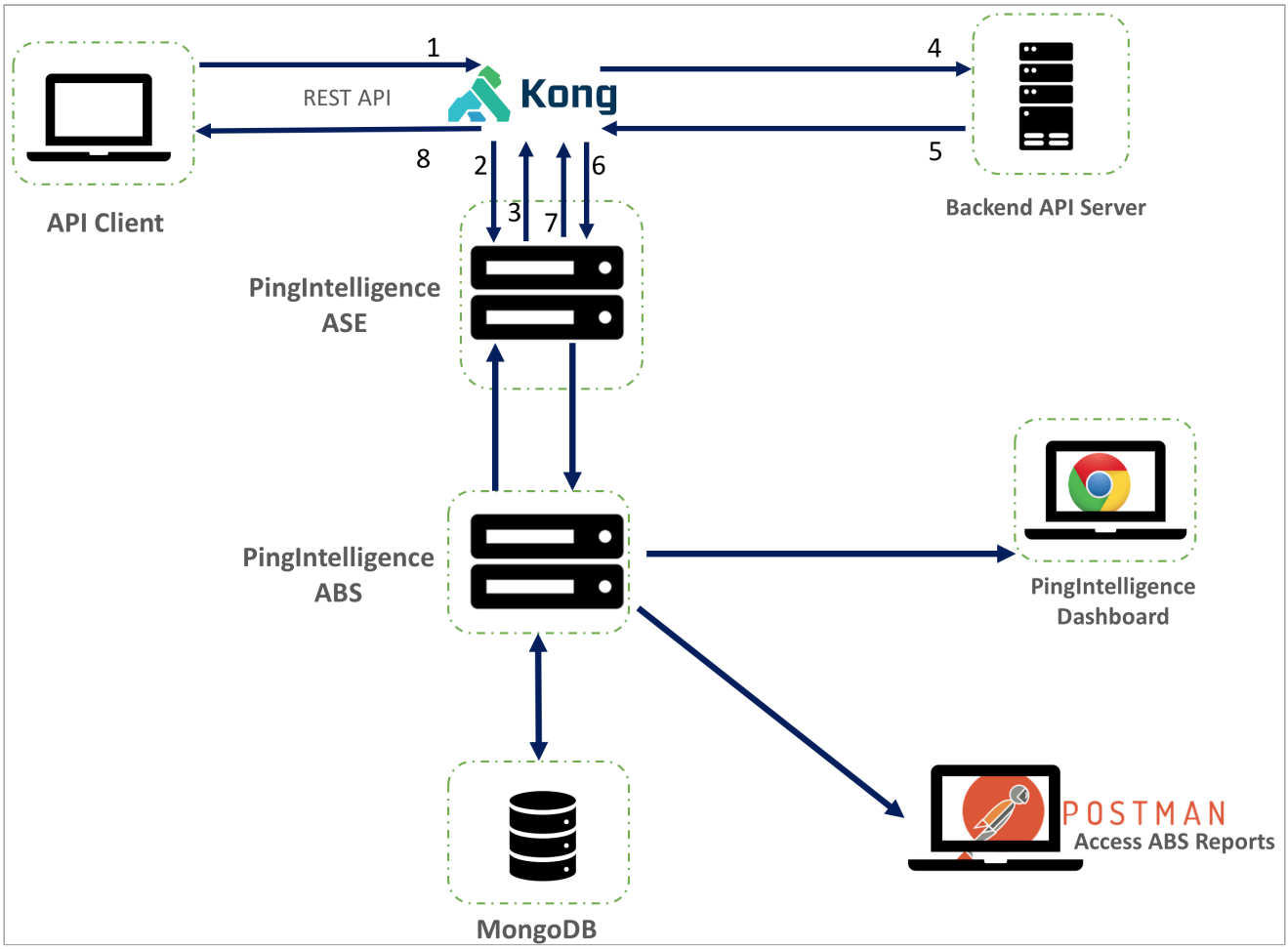
This guide describes the deployment of the PingIntelligence plugin for Kong 1.5.0 community version API gateway.

Install the plugin on all the Kong nodes that you want to integrate with PingIntelligence. You can apply the plugin at the global level or a per-service level for both DB-less and database mode of Kong API gateway. For more information on Kong's DB-less and database mode, see Kong documentation.The following is a high-level list of features of the PingIntelligence plugin:

- You can apply the plugin at the global or per-service level for both database and DB-less mode.
- The plugin supports keep-alive connections.
- You can configure API Security Enforcer (ASE) primary and secondary nodes for failover. If both the primary and secondary nodes are not available, the plugin routes the connection to the backend servers.

The following diagram shows the logical setup of PingIntelligence and Kong API gateway:

Copyright © 2025 Ping Identity Corporation



The following is the traffic flow through Kong API gateway and PingIntelligence components:

1. The client sends an incoming request to Kong.
2. Kong makes an API call to send the request metadata to ASE.
3. ASE checks the request against a registered set of APIs and looks up the client identifier on the PingIntelligence AI engine-generated deny list. If all checks pass, ASE returns a `200-OK` response to Kong. If not, a different response code is sent to Kong. The request information is also logged by ASE and sent to the AI engine for processing.
4. If Kong receives a `200-OK` response from ASE, then it forwards the request to the backend server. A request is blocked only when ASE sends a `403 error` code to Kong.
5. The response from the backend server is received by Kong.
6. Kong makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to Kong.
8. Kong sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

Preparing to deploy the PingIntelligence policy

Complete the following for PingIntelligence and Kong API Gateway before deploying the PingIntelligence plugin.

About this task

Before deploying the PingIntelligence plugin:

Steps

1. Install the PingIntelligence software.

   For more information on installing PingIntelligence for APIs, see Automated deployment guide or Manual deployment guide.

2. Verify that ASE is deployed in `sideband` mode by running the `status` command:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

   ```
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : enabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
   ```

   Troubleshooting:

   If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE. For more information on the `ase.conf` file, see Sideband ASE configuration using the `ase.conf` file.

3. For a secure communication between Kong and ASE, enable sideband authentication by entering the following command in the ASE command line:

   ```
   # ./bin/cli.sh enable_sideband_authentication -u admin –p
   ```

4. To generate the token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use.

   A token is required for Kong to authenticate with ASE. This token is generated in ASE and configured in the `kong.yml` file of the PingIntelligence plugin.

Copyright © 2025 Ping Identity Corporation

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

5. If you want to keep alive the connections beteen Kong and ASE, set the value of `enable_sideband_keepalive` to `true`. If ASE is already running, stop ASE, edit the `ase.conf` file, and then start ASE.

   For more information on keep-alive paramter, see Sideband ASE configuration using the `ase.conf` file.

6. Install the Kong API Gateway and LuaRocks, the Lua package manager, on all the Kong nodes where you want to deploy the PingIntelligence module.

Deploying the PingIntelligence policy

Deploy the PingIntelligence plugin for the Kong API Gateway.

About this task

To deploy the PingIntelligence plugin for Kong API Gateway:

Steps

1. Download the PingIntelligence plugin for Kong and copy to the `/opt/` directory on all the Kong nodes where you want to deploy the PingIntelligence plugin.
2. Untar the plugin file by entering the following command:

   ```
   $ untar pi-api-kong-policy-4.1.0.tar.gz
   ```

3. Change the directory to `/opt/pingidentity/kong-policy`:

   ```
   $ cd /opt/pingidentity/kong-policy
   ```

4. Run the LuaRocks command to deploy the PingIntelligence plugin:

   This command installs the PingIntelligence plugin files at the `/usr/local/share/lua/5.1/kong/plugins/pingintelligence/` location. This location may be different based on the version of LuaRocks.

   ```
   $ luarocks make *.rockspec
   ```

5. Configure `/opt/pingidentity/kong-policy/examples/kong.conf` to provide the plugin name.

   The default plugin name is `pingintelligence`. The plugin name that you configure in `kong.conf` is used in the `kong.yml` file. The following is a sample `kong.conf` file.

   > **Note**
   >
   > Edit your existing `kong.conf` file by copying the `plugins = bundled,pingintelligence` section.

Copyright © 2025 Ping Identity Corporation

```
#------------------------------
# Kong sample configuration file
# ------------------------------

log_level = debug
 plugins = bundled,pingintelligence
proxy_listen = 0.0.0.0:8000
admin_listen = 0.0.0.0:8001
database = off
declarative_config = /opt/pingidentity/kong-policy/examples/kong.yml
lua_ssl_trusted_certificate = /opt/pingidentity/kong-policy/certs/cacert.pem
lua_package_path = ./?.lua;./?/init.lua;
```

6. Optional: If you are running Kong in DB-less mode, configure the `kong.yml` file for deploying the PingIntelligence plugin.

   The following table explains the variables of the file.

Variable	Description
Services	
`name`	The name of the service or API.
`url`	The URL where the service or API is hosted.
`routes`	The subpaths of the service. A maximum of 3-subpaths are supported.
Plugins: Define the ASE specific variables for a service or API.	
`name`	The name of the plugin. This name was configured in the `kong.conf` file.
`service`	The name of the service API. If you want to apply the plugin to more than one service, create a service section for each service as shown in the example `kong.yml` file. For example, if you have three services or APIs, your `kong.yml` file should have three `service` sections, one for each service. The example `kong.yml` file has two sample service names configured.
config	
`ase_primary_host`	IP address of primary ASE node.
`ase_secondary_host`	IP address of the secondary ASE node.
`ase_port`	Port number of the ASE node

Copyright © 2025 Ping Identity Corporation

Variable	Description
`ase_token`	The sideband ASE token that was generated as part of the prerequisites
`ase_timeout`	The time in milliseconds for which Kong waits for ASE to respond before trying the other host. The default value is 5,000 ms
`ase_keepalive`	The time in milliseconds for the keepalive connection. The default value is 60,000 ms.
`access_token`	If OAuth token is part of the query string, the `access_token` field allows you to set the query param key that holds OAuth token in the query string
`use_tls`	Configures a TLS connection between the API gateway and ASE. The default value is `false`.
`sni_name`	Fully qualified domain name (FQDN) of the certificate applied to ASE data port
`tls_verify`	When set to `true`, the API gateway verifies the certificate. If the certificate validation fails, the connection is closed. When set to `false`, the API gateway does not verify the certificate, however, the connection between the API gateway and ASE is encrypted.

1. Per-service level: Configure the `kong.yml` file as described in the table above with the service name of all the API or services to which you want to apply the plugin.

   The following is a sample `kong.yml` file:

Copyright © 2025 Ping Identity Corporation

```
# ------------------------------------------------------------------------------
# This is an example file to get you started with using
# declarative configuration in Kong.
# ------------------------------------------------------------------------------

# Metadata fields start with an underscore (_)
# Fields that do not start with an underscore represent Kong entities and attributes

# _format_version is mandatory,
# it specifies the minimum version of Kong that supports the format

_format_version: "1.1"

# Each Kong entity (core entity or custom entity introduced by a plugin)
# can be listed in the top-level as an array of objects:

services:
  - name: shop-books
    url: <your_service_url>
    routes:
      - name: shop-books-route
        paths:
          - /shopapi-books

  - name: shop-electronics
    url: <your_service_url>
    routes:
      - name: shop-electronics-route
        paths:
          - /shopapi-electronics

plugins:
  - name: pingintelligence
    service: shop-books
    _comment: "An example configuration of pingintelligence plugin"
    config:
      ase_primary_host: localhost
      ase_secondary_host: localhost

      ase_port: "8000"
      ase_token: 1ebd5fde1b0b4373a1ad8b8724d13813
      ase_timeout: "5000"
      ase_keepalive: "60000"
      access_token: access_token
      use_tls: false
      sni_name: test.ase.pi
      tls_verify: false
    tags:
      - api_security

  - name: pingintelligence
    service: shop-electronics
    _comment: "An example configuration of pingintelligence plugin"
    config:
```


Copyright © 2025 Ping Identity Corporation

```
      ase_primary_host: 172.16.40.220
      ase_secondary_host: 172.16.40.220
      ase_port: "8000"
      ase_token: 1ebd5fde1b0b4373a1ad8b8724d13813
      ase_timeout: "5000"
      ase_keepalive: "60000"
      access_token: access_token
      use_tls: false
      sni_name: test.ase.pi
      tls_verify: false
    tags:
      - api_security
```

2. Global level: To apply the plugin at the global level, remove the `service` name from the `kong.yml` file as shown in the sample file below:

Copyright © 2025 Ping Identity Corporation

```
# ------------------------------------------------------------------------------
# This is an example file to get you started with using
# declarative configuration in Kong.
# ------------------------------------------------------------------------------

# Metadata fields start with an underscore (_)
# Fields that do not start with an underscore represent Kong entities and attributes

# _format_version is mandatory,
# it specifies the minimum version of Kong that supports the format

_format_version: "1.1"

# Each Kong entity (core entity or custom entity introduced by a plugin)
# can be listed in the top-level as an array of objects:

services:
    url: <your_service_url>
    routes:
      paths:


plugins:
  - name: pingintelligence
     _comment: "An example configuration of pingintelligence plugin"
    config:
      ase_primary_host: localhost
      ase_secondary_host: localhost

      ase_port: "8000"
      ase_token: 1ebd5fde1b0b4373a1ad8b8724d13813
      ase_timeout: "5000"
      ase_keepalive: "60000"
      access_token: access_token
      use_tls: false
      sni_name: test.ase.pi
      tls_verify: false
    tags:
      - api_security
```

7. Start the API gateway after the plugin has been deployed:

```
$ kong start -c kong.conf
```

> **Note**
>
> By default, Kong is configured to run its services on 8000 port and admin API on 8001 port. You can change these default ports in the `kong.conf` file.

8. Optional: Configure Kong to work in database mode. If you are running Kong in database mode, use the following `curl` commands to apply the plugin at a per-service level or global level.

Copyright © 2025 Ping Identity Corporation

You can refer the config section in step 6 above for more details on the parameters sent as part of the request in the `curl` commands. Make sure that Kong is running when you are applying the plugin in database mode.

Choose from:

- Service level: Run the following command to apply the plugin at a per service level:

```
curl --location --request POST '<kong_ip>:<kong_admin_port>/services/<service_name>/plugins' \
--header 'Content-Type: application/json' \
--data-raw '{
        "name": "pingintelligence",
    "config": {
        "tls_verify": ,
        "sni_name": "",
        "ase_port": "",
        "ase_primary_host": "",
        "ase_token": "",
        "ase_timeout": "",
        "ase_keepalive": "",
        "ase_secondary_host": "",
        "access_token": "",
        "use_tls":
    }
}'
```

- Global level: Run the following `curl` command to apply the plugin at the global level:

```
curl --location --request POST '<kong_ip>:<kong_admin_port>/plugins' \
--header 'Content-Type: application/json' \
--data-raw '{
        "name": "pingintelligence",
      "config": {
        "tls_verify": ,
        "sni_name": "",
        "ase_port": "",
        "ase_primary_host": "",
        "ase_token": "",
        "ase_timeout": "",
        "ase_keepalive": "",
        "ase_secondary_host": "",
        "access_token": "",
        "use_tls":
    }
}'
```

Extracting user information when an OIDC plugin is installed

Extract user attributes from JavaScript Object Notation (JSON) web tokens (JWT) when an OpenID Connect (OIDC) plugin is installed in the Kong Gateway.

About this task

To extract user attributes:

Copyright © 2025 Ping Identity Corporation

Steps

1. Capture the header value assigned to the `upstream_introspection_header` parameter in the OIDC plugin configuration.
2. Assign the header value to the `location` key in the `jwt` object of the API JSON file.

   Result:

   API Security Enforcer (ASE) will extract the user information from the JWT.
3. If `upstream_introspection_header` is not configured in the OIDC plugin, then complete the following configuration and assign `x_introspection` to the `location` key in the `jwt` object of the API JSON file:

   ```
   http patch  :8001/plugins/$PLUGIN_ID config:=@patch.json
   cat patch.json
   {
     "upstream_introspection_header":  "x_introspection"
   }
   ```

   Example:

   The following is a snippet of JWT object from a sample API JSON file:

   ```
   "jwt": {
    "location": "h:x_introspection",
   "username": "username",
   "clientid": "client_id"
   }
   ```

Next steps

For more information on configuring the API JSON file, see Defining an API using API JSON configuration file in sideband mode.

MuleSoft sideband integration

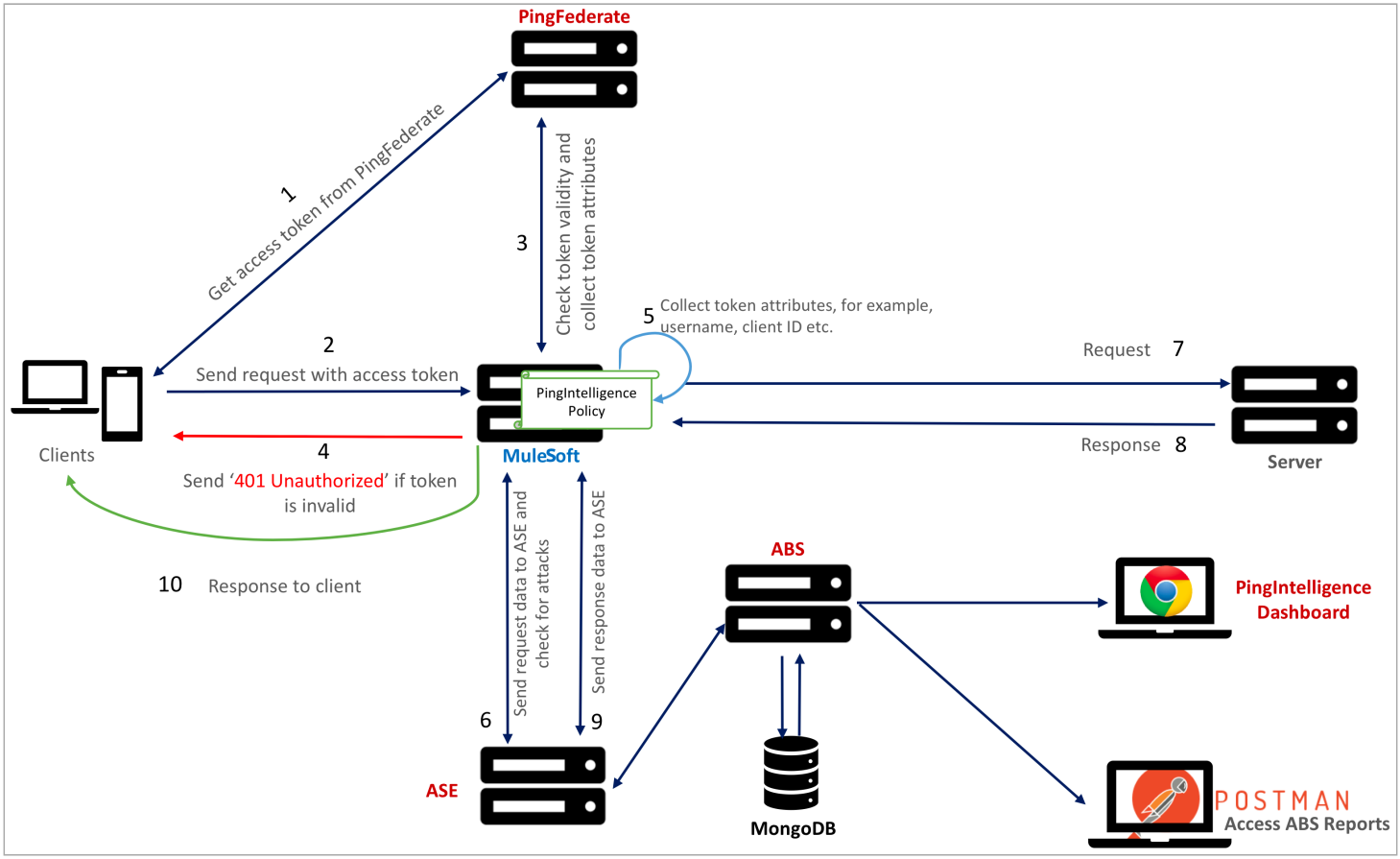
This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with MuleSoft API Gateway.

A PingIntelligence policy is installed in the MuleSoft API Gateway and it passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.

The PingIntelligence policy works with APIs that are configured with basic endpoint and also with APIs that are configured with proxy endpoint. The policy is simpler to deploy when applied to APIs that are configured with the endpoint with proxy option since more API metadata is already accessible by the policy.

Copyright © 2025 Ping Identity Corporation

Traffic flow for MuleSoft integration without user information



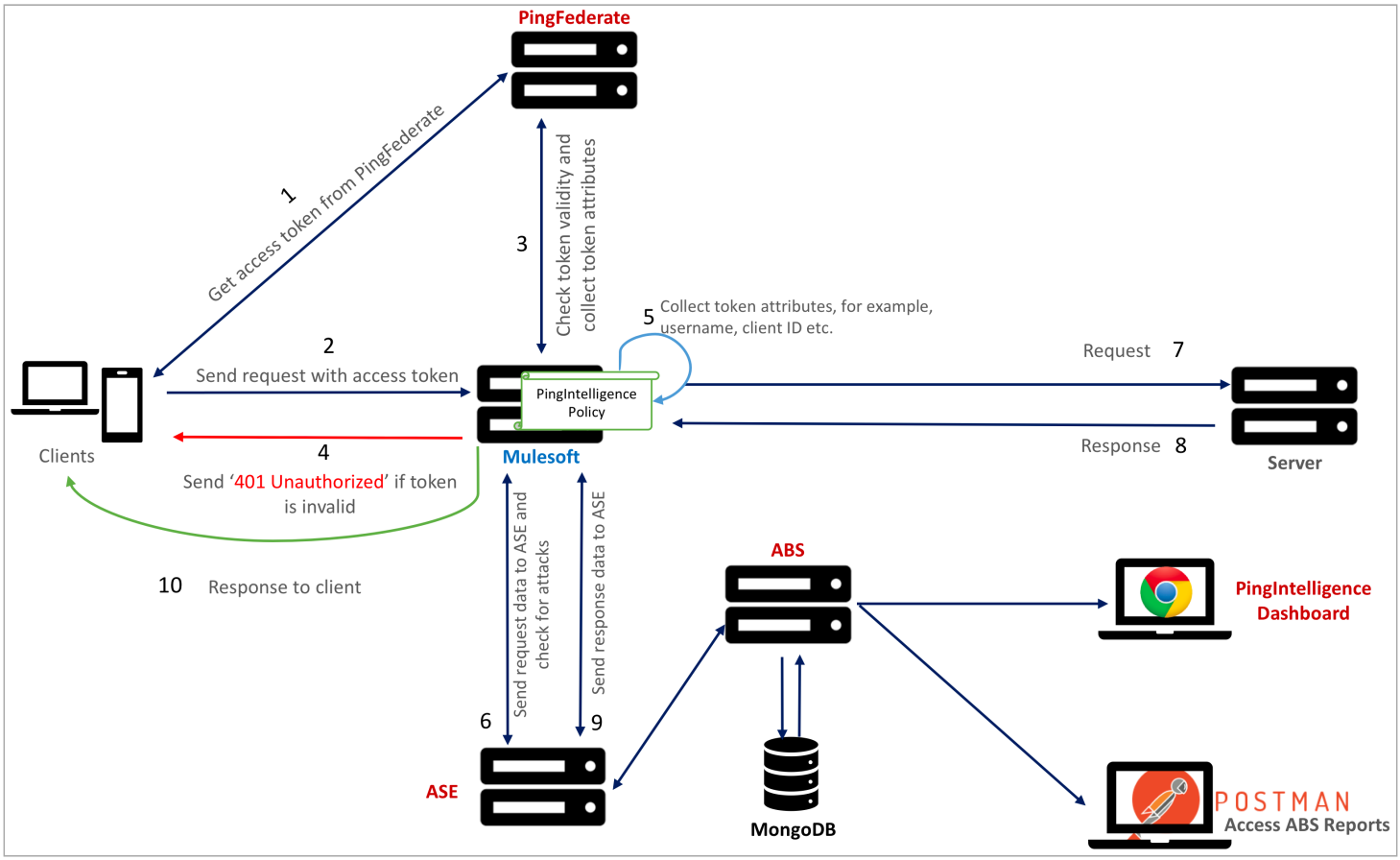
The following is the traffic flow through the MuleSoft and PingIntelligence for APIs components:

1. The client sends an incoming request to MuleSoft.
2. The PingIntelligence policy running in MuleSoft collects API metadata and token attributes.
3. MuleSoft makes an API call to send the request information to ASE. ASE checks the request against a registered set of APIs and checks the origin IP, cookie, or OAuth2 token against the AI-generated deny list. If all checks pass, ASE returns a `200-OK` response to the MuleSoft. If not, a different response code is sent to MuleSoft. The request information is also logged by ASE and sent to the AI engine for processing.
4. If MuleSoft receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, the Gateway optionally blocks the client.
5. The response from the backend server is received by MuleSoft. MuleSoft sends the response received from the backend server to the client.
6. MuleSoft makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing. ASE receives the response information and sends a `200-OK` to MuleSoft.
7. MuleSoft sends the response to the client.

Copyright © 2025 Ping Identity Corporation

Traffic flow for MuleSoft integration with user information

The following is the traffic flow through the MuleSoft and PingIntelligence for APIs components. PingFederate is used as the OAuth server to gather the user information.



1. The client requests and receives an access token from PingFederate.
2. The client sends a request with the access token received from PingFederate.
3. MuleSoft verifies the authenticity of the access token with PingFederate.
4. If the token is invalid, MuleSoft returns a `401-unauthorized` message to the client.
5. If the token is valid, the PingIntelligence policy running in MuleSoft collects API metadata and token attributes.
6. MuleSoft makes an API call to send the request information to ASE. ASE checks the request against a registered set of APIs and checks the origin IP, cookie, or OAuth2 token against the artificial intelligence (AI)-generated deny list. If all checks pass, ASE returns a `200-OK` response to the MuleSoft. If not, a different response code is sent to MuleSoft. The request information is also logged by ASE and sent to the AI engine for processing.
7. If MuleSoft receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, the Gateway optionally blocks the client.
8. The response from the backend server is received by MuleSoft. MuleSoft sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

9. MuleSoft makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing. ASE receives the response information and sends a `200-OK` to MuleSoft.
10. MuleSoft sends the response to the client.

Preparing to deploy the PingIntelligence policy

Complete the following prerequisites before deploying the PingIntelligence policy on MuleSoft.

About this task

Before deploying the PingIntelligence policy:

Steps

1. Verify that MuleSoft version 3.9.x or 4.x is installed.

   If you are using any other version, contact Ping Identity support.

   > **Note**
   >
   > Due to a known bug in MuleSoft 4.2.2, you can encounter a `502` error response when the PingIntelligence policy is deployed with MuleSoft 4.2.2. Refer to thehttps://help.mulesoft.com/s/article/Scatter-Gather-throwing-Event-instance-or-a-MessagingException-on-4-2-2-only[MuleSoft documentation] for more information about the issue and its resolution.

2. Install and configure the PingIntelligence software.

   For more information, see PingIntelligence manual deployment or PingIntelligence automated deployment.

3. Verify that API Security Enforcer (ASE) is in `sideband` mode by running the following ASE command:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Result:

   ```
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : disabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.61 MB, free 102.39 MB
   google pubsub           : disabled
   log level               : debug
   timezone                : local (UTC)
   ```

   Troubleshooting:

Copyright © 2025 Ping Identity Corporation

If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

4. For a secure communication between MuleSoft Anypoint and ASE, enable sideband authentication by entering the following ASE command:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

5. To generate the token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use:

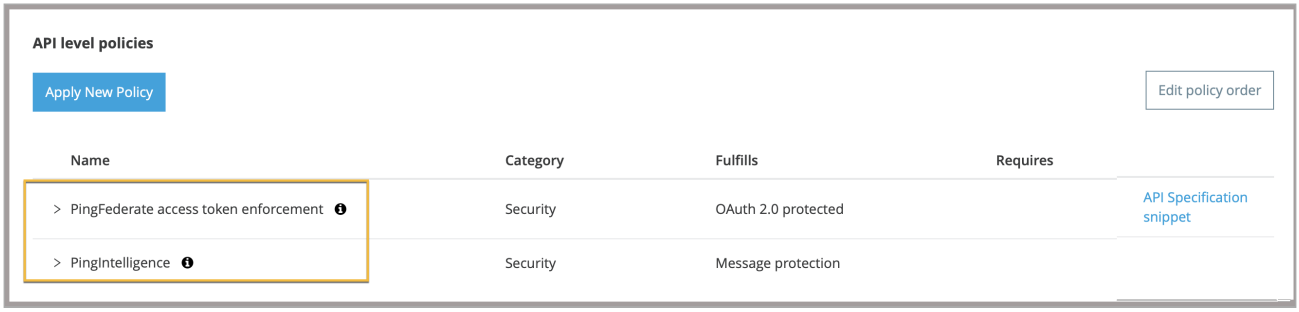
   A token is required for MuleSoft Anypoint to authenticate with ASE.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

6. Optional: Gather user information from PingFederate:

   1. To integrate PingFederate with MuleSoft, follow the instructions in Configure Client Management PingFederate.

      This will enable PingFederate OAuth Token Enforcement policy. This policy should be applied before the PingIntelligence policy in the Anypoint platform API Manager as shown in the following screenshot.



   Currently the PingIntelligence policy supports PingFederate as authorization server.

Deploying the PingIntelligence policy

The PingIntelligence sideband policy supports integration with MuleSoft 3.9 and 4.x API Gateways.

Before you begin

The policy package has the following two files:

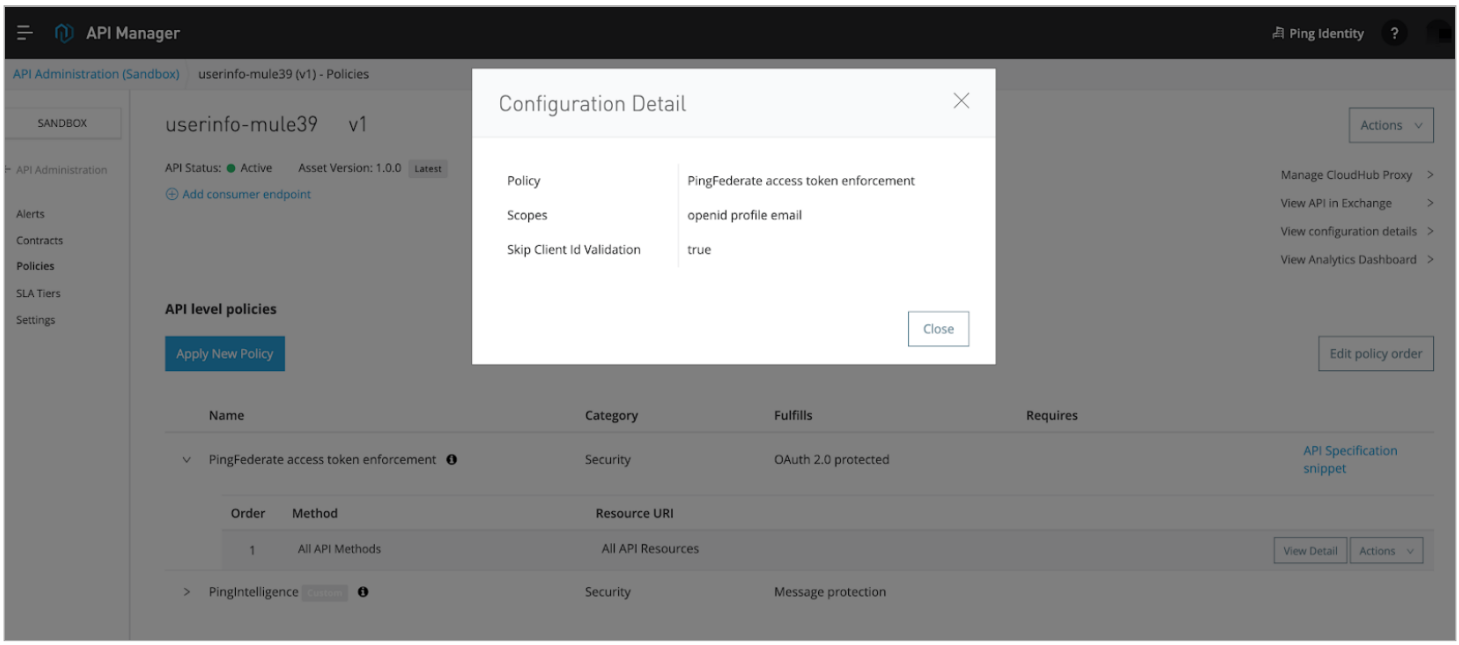
Steps

- `policy.yaml`
- `policy.xml`

Next steps

Copyright © 2025 Ping Identity Corporation

Follow the steps to deploy PingIntelligence policy based on the version of the MuleSoft API Gateway. For PingIntelligence to detect attacks based on username, make sure that the PingFederate access token enforcement policy is the first policy deployed. PingIntelligence policy should be the second policy.



MuleSoft 3.9

Deploying PingIntelligence for MuleSoft 3.9

About this task

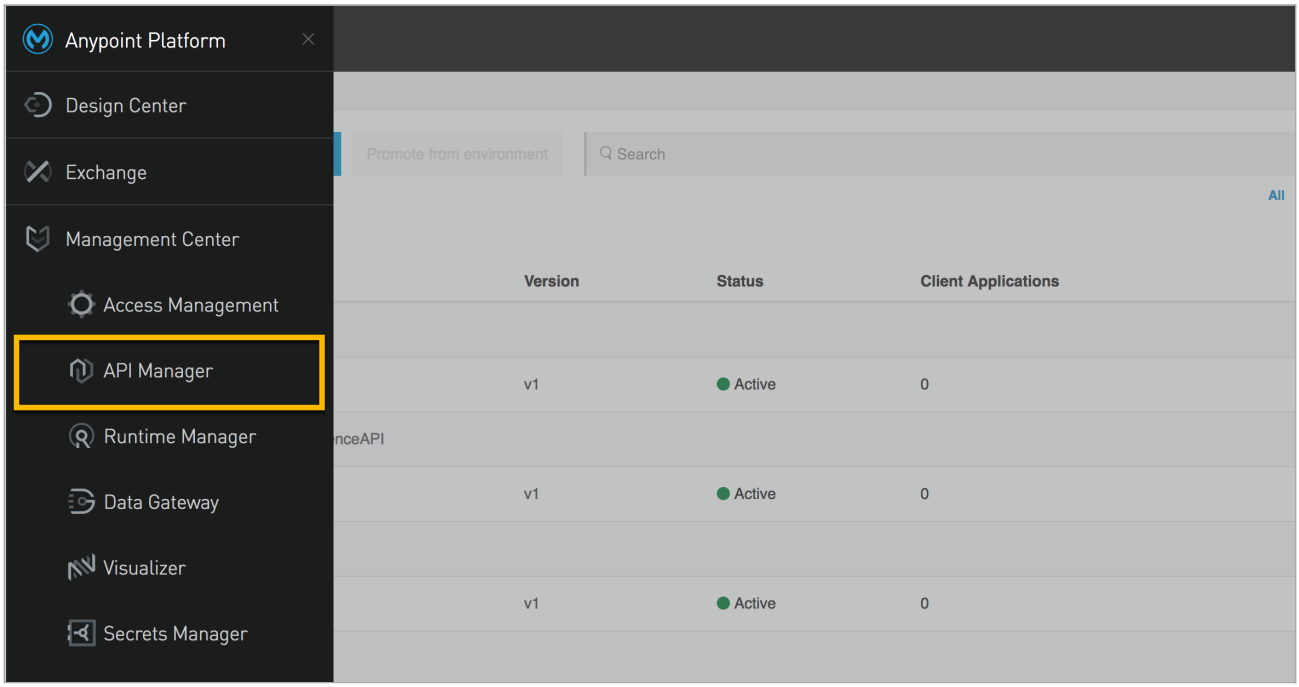
Before applying the PingIntelligence policy, make sure that the API to which you want to apply the policy is defined. The steps below use an API named PingIntelligenceAPI for illustration purposes.

To deploying the PingIntelligence policy to MuleSoft Anypoint 3.9:

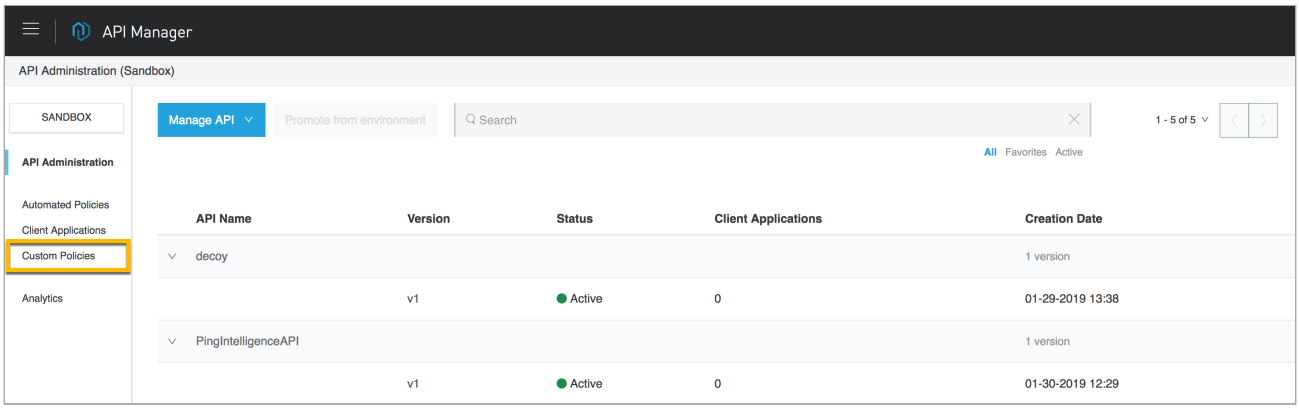
Steps

1. **Sign on to your MuleSoft Anypoint account.**
2. **Open API Manager by expanding the menu on the left-hand side.**

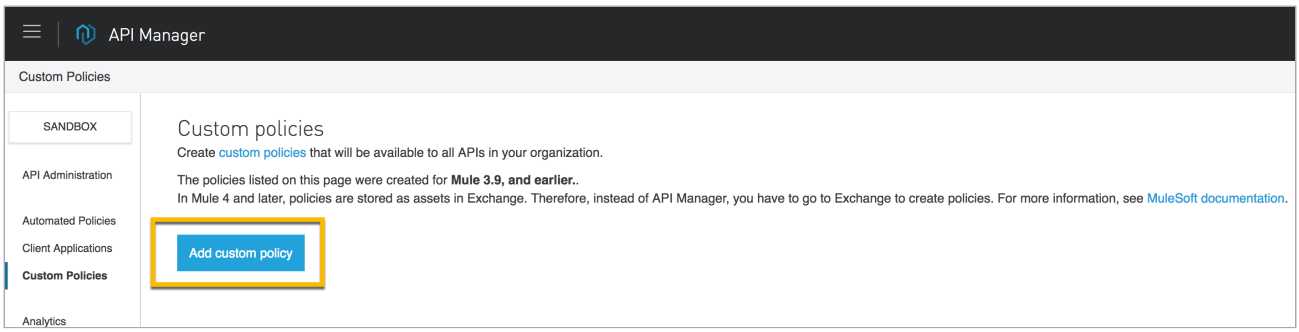
Copyright © 2025 Ping Identity Corporation



3. In the API Administration page, click Custom Policies.

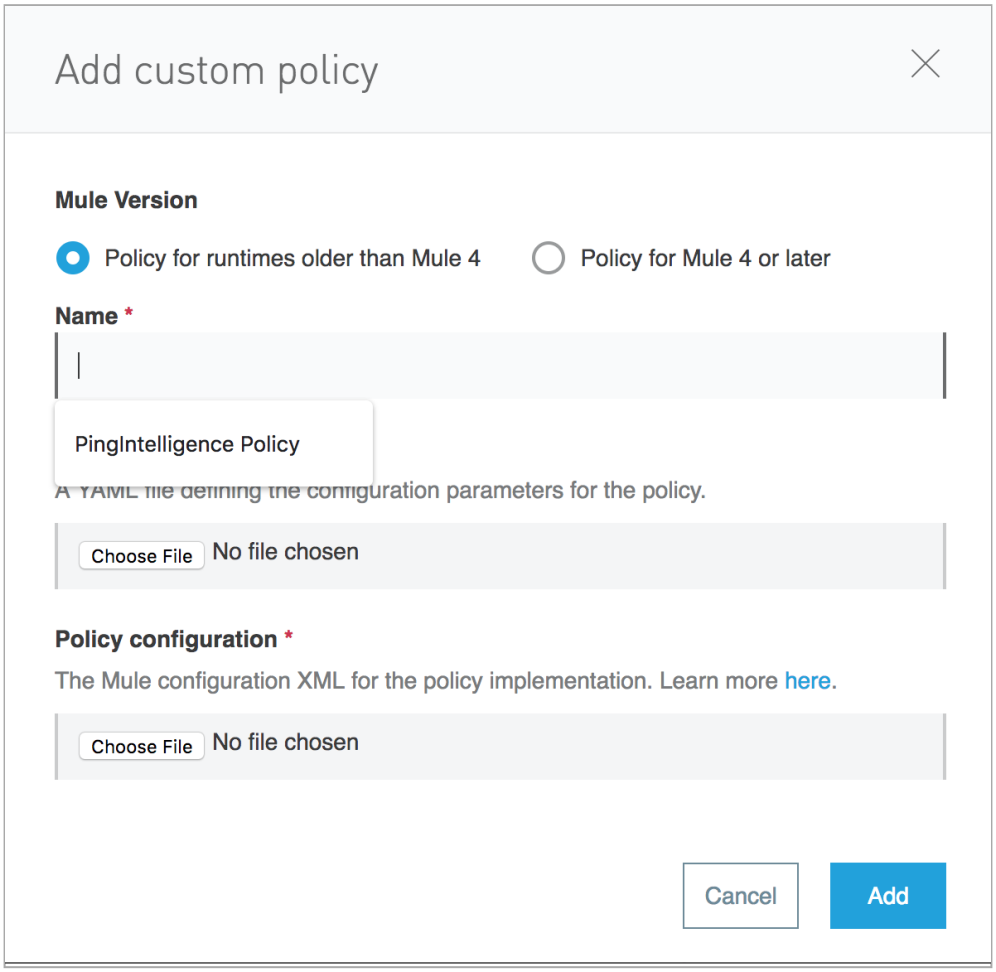


4. In the Custom Policies page, click Add custom policy:



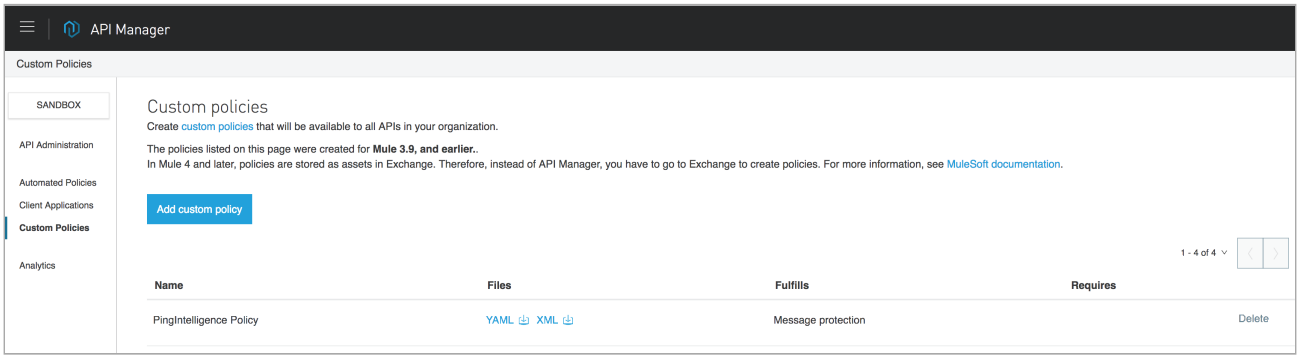
5. In the Add custom policy pop-up window, add the policy name (for example, `PingIntelligence policy`) and upload the `policy.yaml` and `policy.xml` files.

Copyright © 2025 Ping Identity Corporation



Result:

The PingIntelligence policy is added as shown below.



MuleSoft 4.x

Deploying PingIntelligence for MuleSoft 4.x

About this task

The PingIntelligence policy for MuleSoft now supports two languages:

- **Groovy**
- **Java**

Based on your environment and requirements, either of the policies can be deployed.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> The PingIntelligence MuleSoft Java policy supports asynchronous invocation from the Gateway to PingIntelligence ASE.

The PingIntelligence policy for MuleSoft tar is now comprised of the following structure (after extraction):

```
pingidentity/
`-- mulesoft-policy
    |-- mulesoft-3.9
    |   |-- policy.xml
    |   `-- policy.yaml
    |-- mulesoft-4.0-groovy
    |   |-- policy.xml
    |   |-- policy.yaml
    |   `-- pom.xml
    `-- mulesoft-4.0-java
        |-- mule-artifact.json
        |-- policy.xml
        |-- policy.yaml
        `-- pom.xml
```

Complete the following steps to deploy the PingIntelligence policy for the MuleSoft 4.0 Groovy or Java policy:

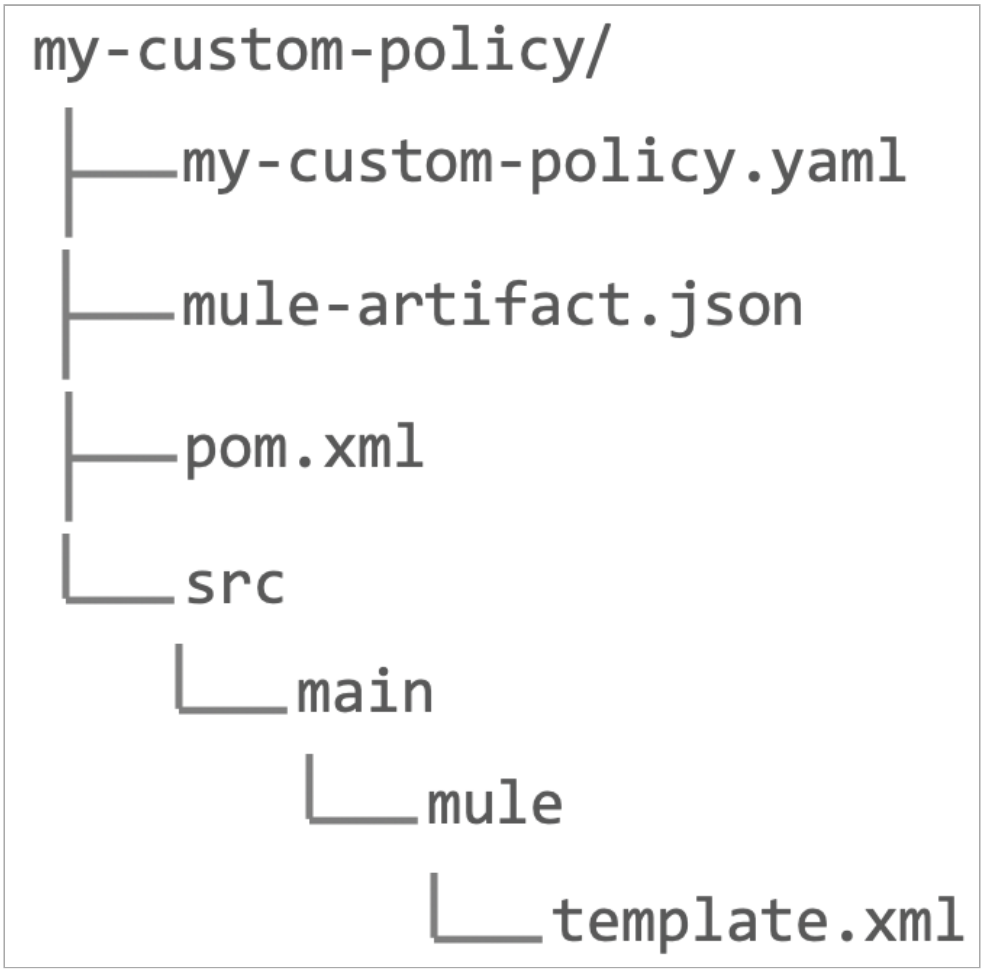
To deploy the PingIntelligence policy for the MuleSoft 4.x or MuleSoft 4.0 Groovy or Java policy:

Steps

1. **Create a project directory by following the instructions in Getting started with Custom Policies development.**

The following screenshot shows an illustrative sample of a project directory structure.

Copyright © 2025 Ping Identity Corporation

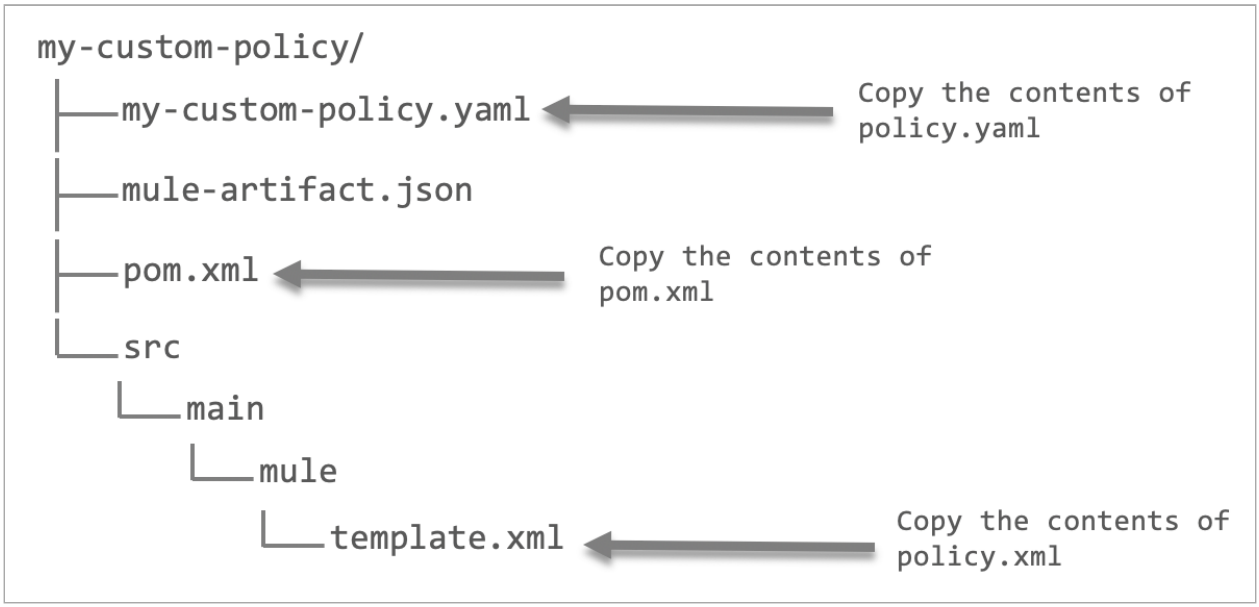


+ The PingIntelligence policy package provides three files for 4.x:

- `policy.xml` : Contains the actual logic of the policy.
- `policy.yaml` : Has details that render policy configuration UI.
- `pom.xml` : Specifies dependencies for policy compilation.

1. When the project's directory structure is created, replace the contents of `my-custom-policy.yaml` with that of the `policy.yaml` file, and the contents of `template.xml` with that of `policy.xml`. Similarly, replace the contents of `pom.xml` with that of the `pom.xml` file provided in the PingIntelligence policy package.

Copyright © 2025 Ping Identity Corporation



2. Edit the `pom.xml` file to enter your organization's `groupID`:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

     <groupId>7a0f5884-ba26-4929-a681-66ca288a6992</groupId>
    <artifactId>PingIntelligence</artifactId>
    <version>1.2.0</version>
    <name>PingIntelligence</name>
    <description>ASE Sideband Policy for mule 4 with username info</description>
```

3. From the command line in your project folder, run the following command.

   This packages the PingIntelligence policy and creates a deployable JAR file.

```
> mvn clean install
```

> **Note**
>
> You require license to MuleSoft Enterprise Repository for compiling the policy.

4. Upload the PingIntelligence policy to Exchange by following the instructions in Deploying a Policy Created Using the Maven Archetype.

   Result:

The PingIntelligence policy is now available to apply to your APIs. For more information, see Applying the PingIntelligence policy.

Copyright © 2025 Ping Identity Corporation

Applying the PingIntelligence policy

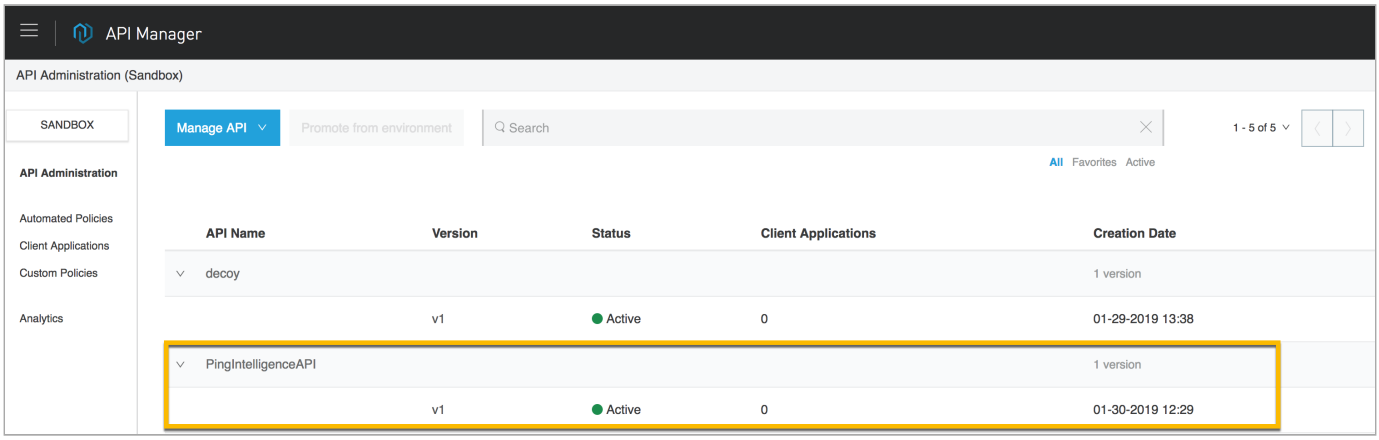
Complete the following steps to attach the PingIntelligence policy to your API.

About this task

> **Note**
>
> If you are applying the PingIntelligence policy in MuleSoft 3.9 and there is an earlier version of the policy already applied to your API, then remove the policy before applying the PingIntelligence 4.3 policy. To remove the policy, follow the steps in Removing an existing PingIntelligence policy.

Steps

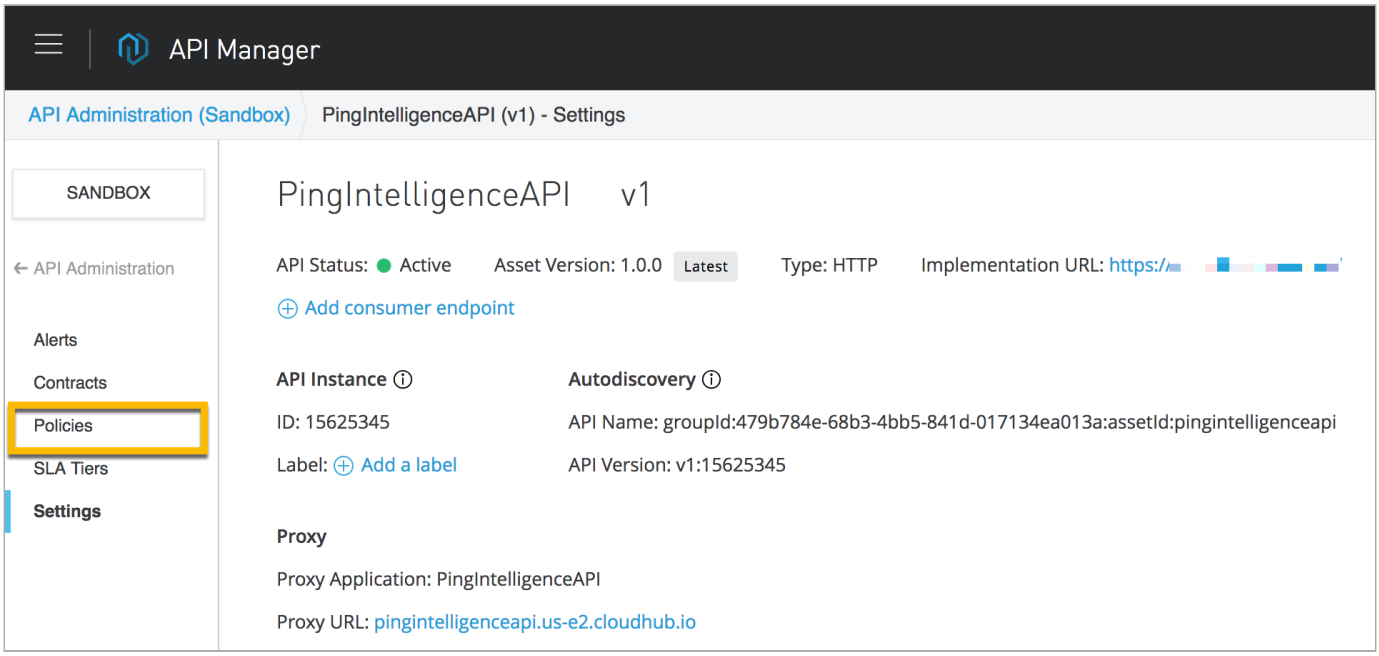
1. **Sign on to your MuleSoft Anypoint account.**
2. **Navigate to the API Manager and click the Version of the API to which you want to attach the PingIntelligence policy.**



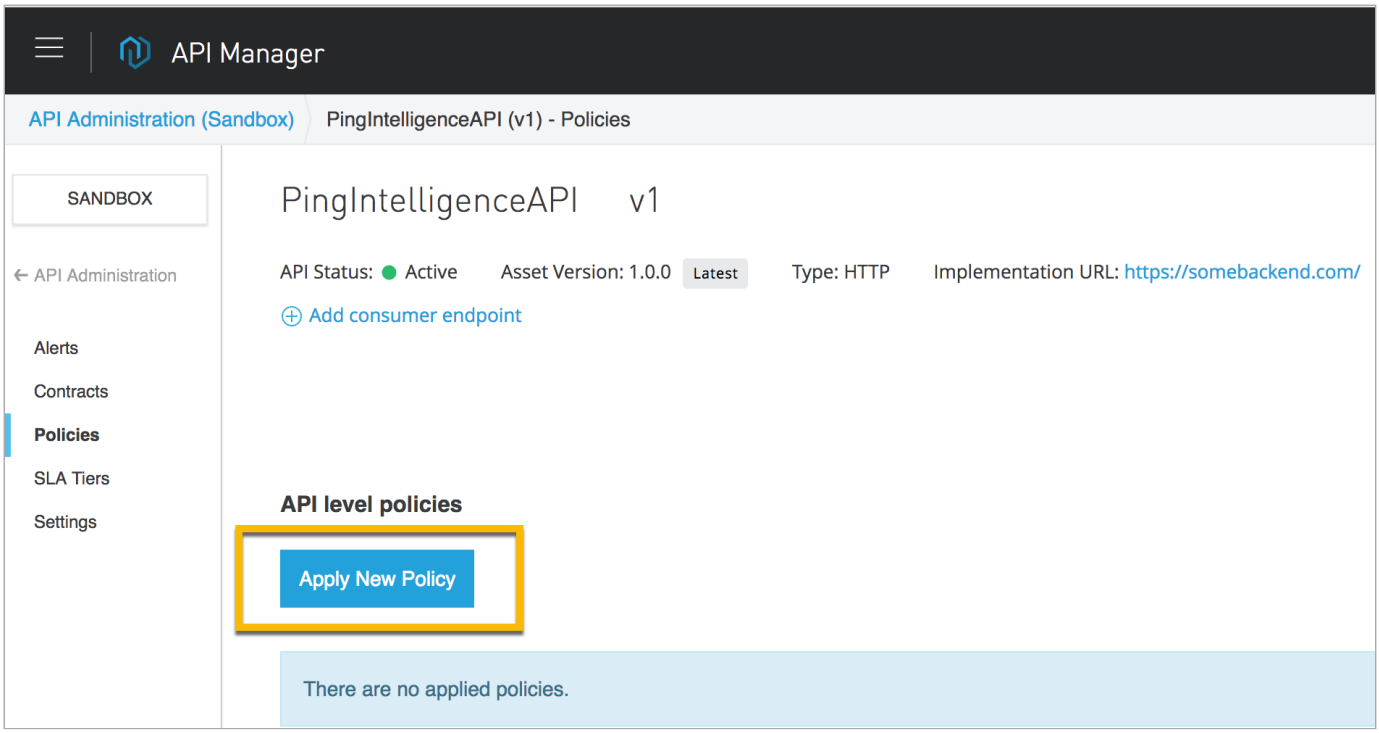
3. **On the API page, click Policies.**

   The Policies page supports applying the PingIntelligence policy to the API.

Copyright © 2025 Ping Identity Corporation

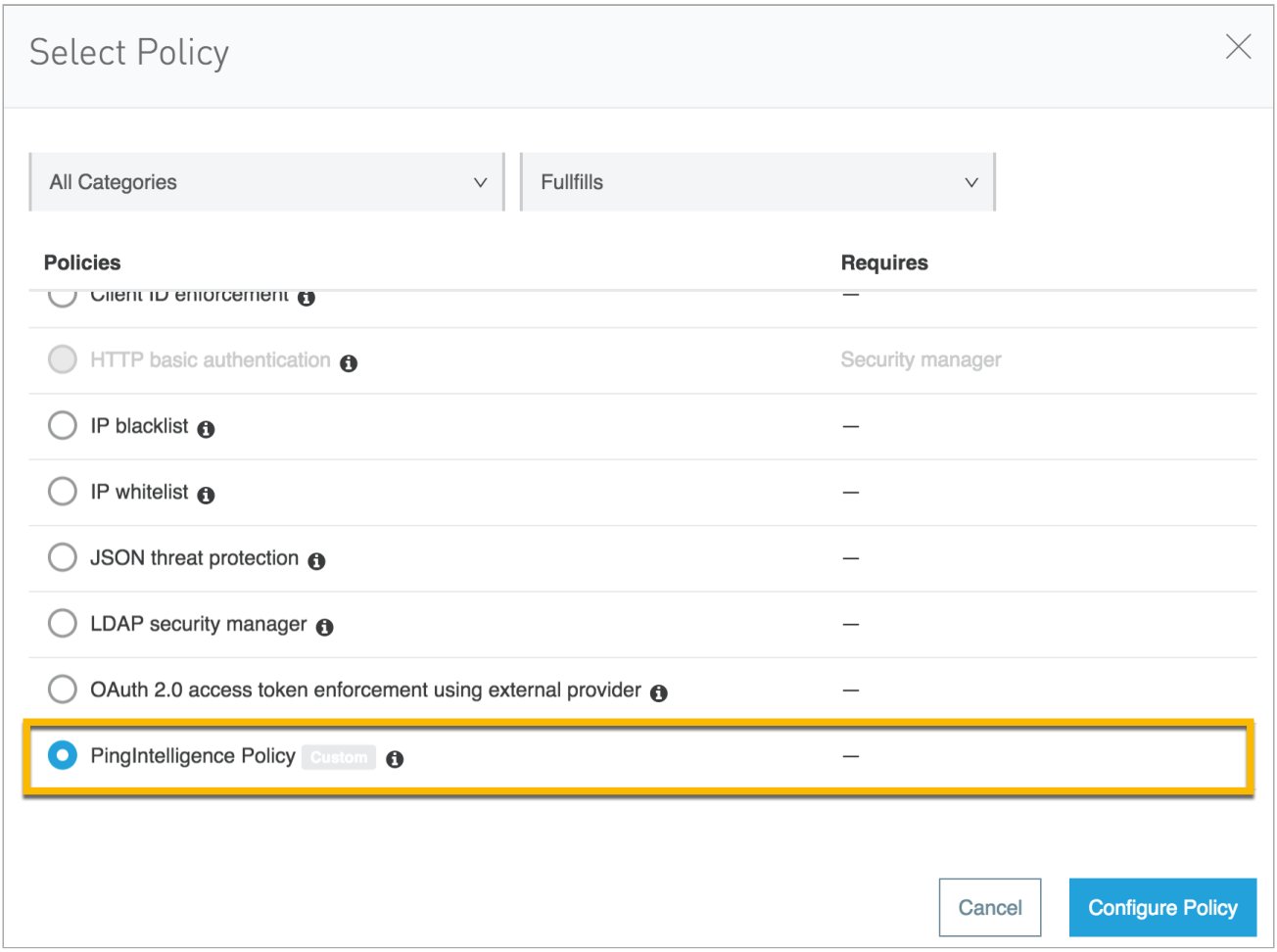


4. Click Apply New Policy.



5. In the Select Policy pop-up window, select the PingIntelligence policy and click Configure Policy.

Copyright © 2025 Ping Identity Corporation



6. In the Apply policy page, enter the following values:
   - ASE Token that was generated as part of prerequisite.
   - ASE primary and secondary host and port. The traffic is sent to the ASE secondary host only when the primary ASE node is unreachable.
   - Enable SSL for a secure HTTPS connection between Mulesoft and PingIntelligence ASE.
   - Check the Allow self-signed certificate check-box to enable Mulesoft to accept a self-signed certificate from ASE.
   - If the Allow asynchronous (non-blocking) request forwarding check box is selected, the API Gateway will not wait for a response from ASE before propagating the inbound request.

     Note

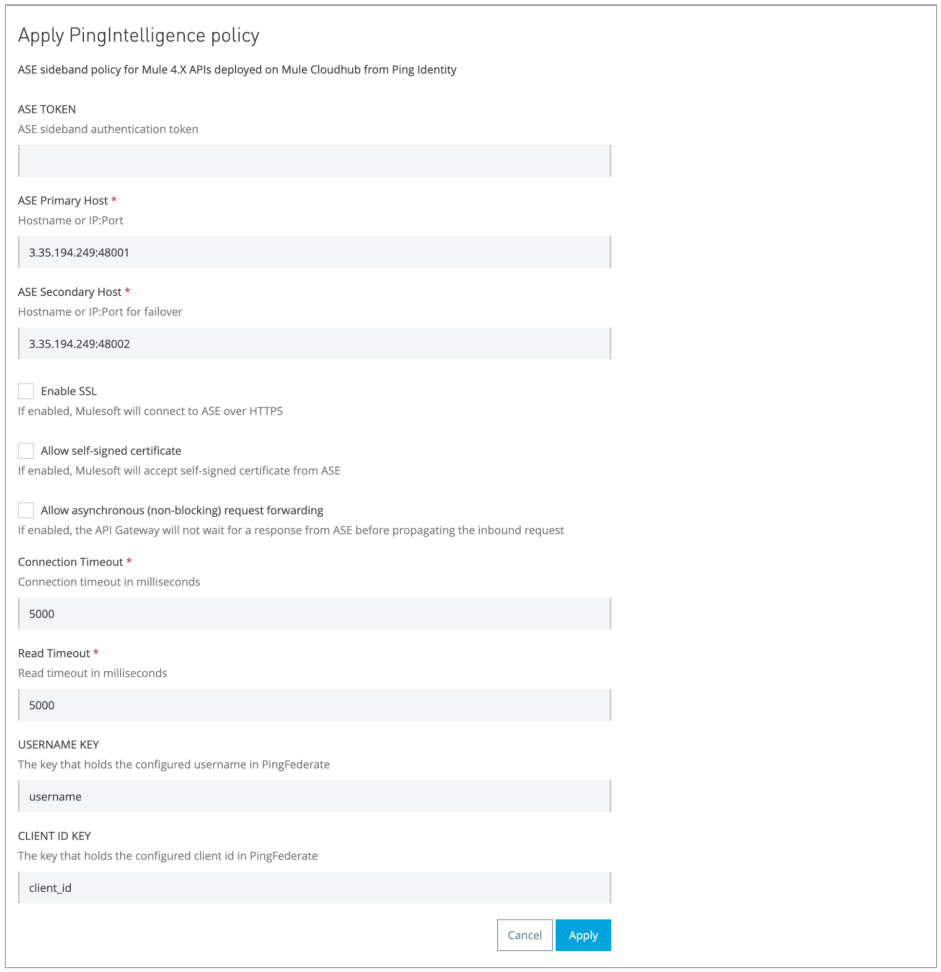
     The asynchronous option is only available in the PingIntelligence MuleSoft Java policy.

   - Configure the Connection Timeout and Read Timeout. The behavior of the API gateway is governed by Connection Timeout and Read Timeout, in the event of API Gateway not able to connect to ASE or the response from ASE is delayed.

Copyright © 2025 Ping Identity Corporation

Timeout parameter	Description
Connection Timeout	It governs the time the API gateway waits to establish a connection with ASE, following which it sends the client request to the backend server.
Read Timeout	It governs the time the API Gateway waits for ASE's response before sending the request to the backend server.

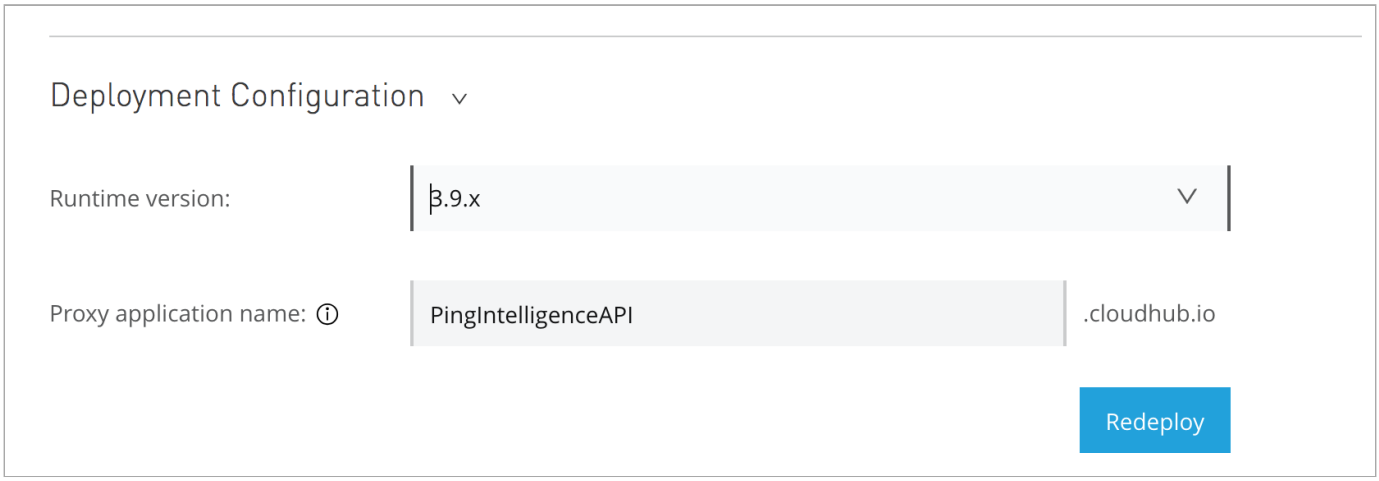
The default value is 5000 milliseconds or 5 seconds. It is good practice to configure a small value to limit the delay in case ASE is not reachable or unresponsive.



> **Note**
>
> If there are any changes to the ASE endpoints, repeat the process explained in step 6 and re-deploy the configuration.

7. **Navigate to your API and click the version number as described in step 1. In the API page, scroll down to the Deployment Configuration section and click Redeploy.**

Copyright © 2025 Ping Identity Corporation



8. If your API is configured with Basic endpoint on MuleSoft version 3.9.x, then add the following properties in your MuleSoft application:
    - `http.status`
    - `http.reason`
    - `content-type`
    - `content-length`

    You can use the `set-property` element to configure these properties in the MuleSoft application. If required, you can also set other response side headers to send more information to the PingIntelligence policy.

    +

    Result:

    The following is a sample configuration of setting response side details. For more information on setting the properties in a Mule application, see property transformer.

```
<set-property propertyName="http.status" value="200" doc:name="Property"/>
<set-property propertyName="http.reason" value="OK" doc:name="Property"/>
<set-property propertyName="content-type" value="application/json" doc:name="Property"/>
<set-property propertyName="content-length" value="21" doc:name="Property"/>
<set-property propertyName="set-cookie" value="PHPSESSIONID=CookieValue" doc:name="Property"/>
```

Removing an existing PingIntelligence policy

Remove earlier versions of PingIntelligence policy applied on your APIs before applying the PingIntelligence 4.3 policy.

About this task

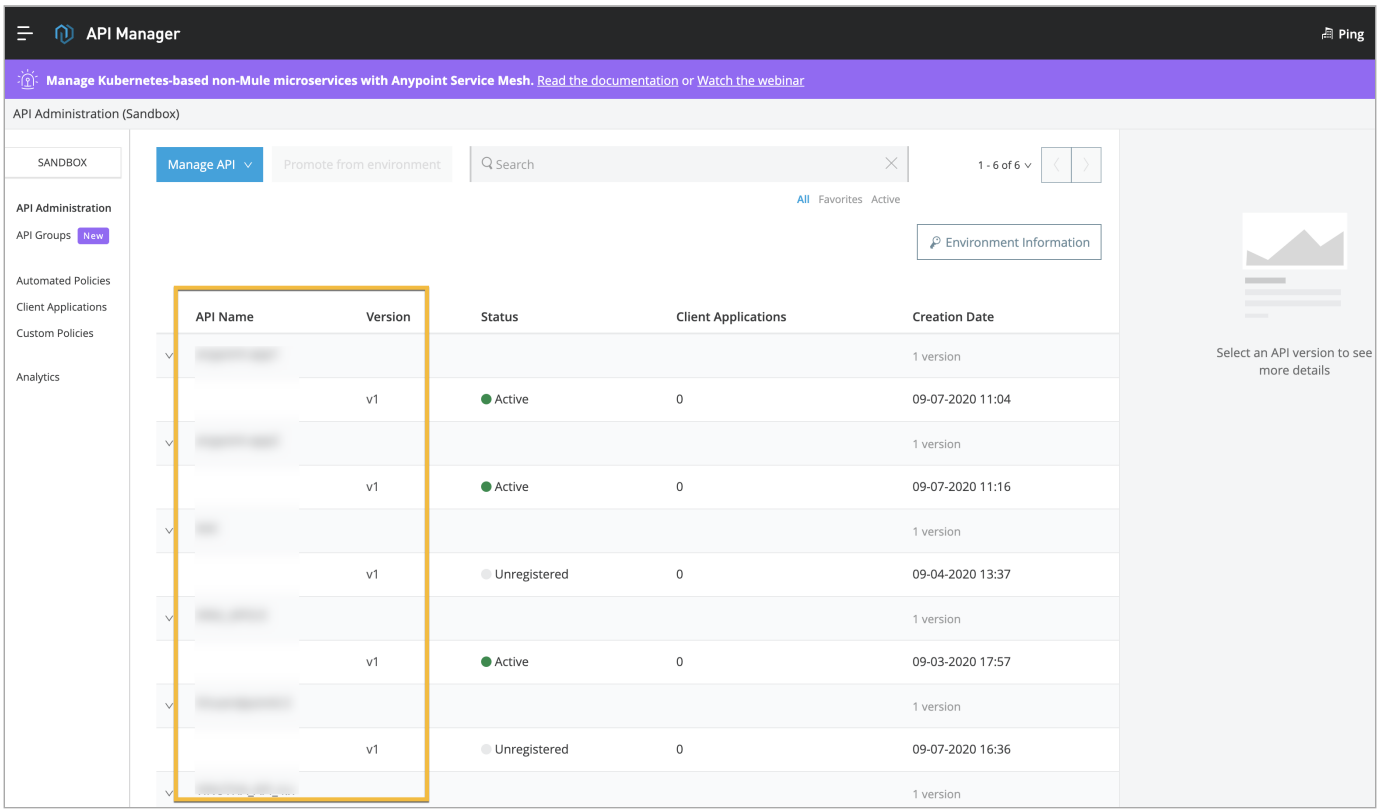
To remove an exising policy from your API:

Steps

1. Sign on to your MuleSoft Anypoint account.

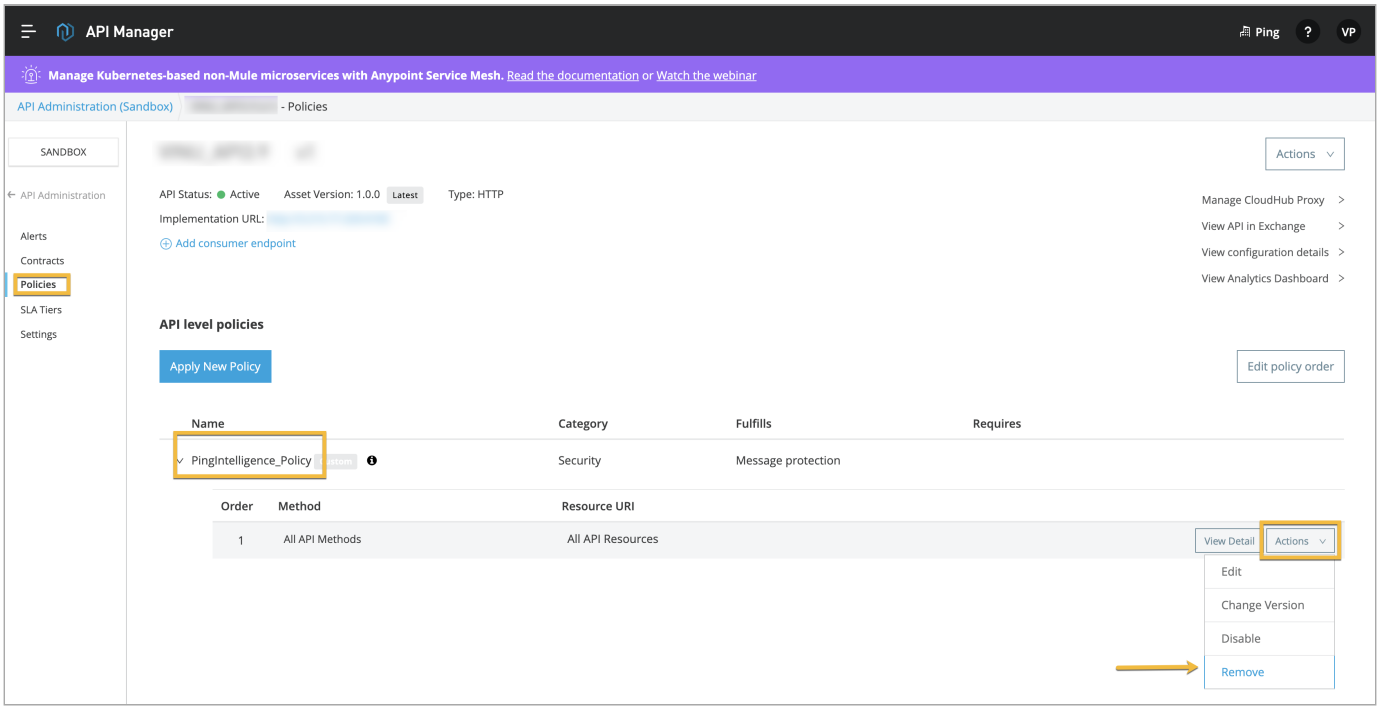
Copyright © 2025 Ping Identity Corporation

2. Navigate to the API Manager. On the API Administation page, click on Version of the API for which you want to remove the PingIntelligence policy.



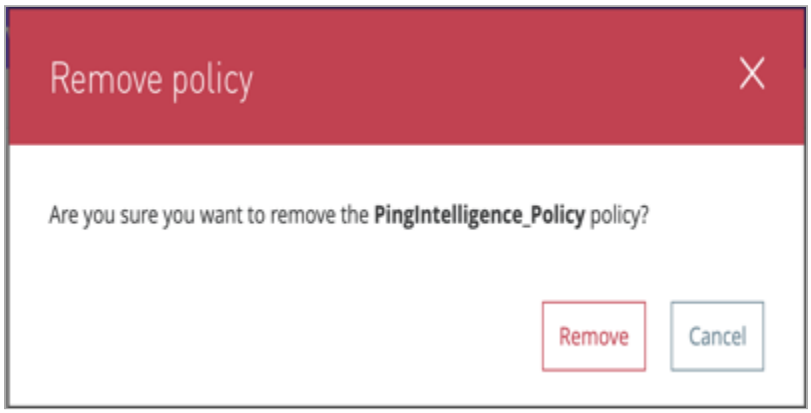
3. On the API page, click Policies, and then click to expand the PingIntelligence policy.

4. From the Actions list, select Remove.



Copyright © 2025 Ping Identity Corporation

5. Click Remove to confirm the policy removal.



Next steps

Once you remove the existing policy from the API, follow the steps in Applying the PingIntelligence policy and apply the new PingIntelligence 4.3 policy.

Integrating PingIntelligence

After deploying the policy, integrate PingIntelligence.

Before you begin

Refer to the ASE and ABS AI Engine Admin Guides.

About this task

To integrate PingIntelligence:

Steps

1. Refer to the following:
    - ASE ports
    - API naming guidelines
    - Adding APIs in Defining an API using API JSON configuration file in sideband mode.

    You can add individual APIs or you can configure a global API. For more information, seeAPI naming guidelines and API discovery and configuration.
    - Connect ASE and ABS
2. After you have added your APIs in ASE, train the API model. Refer to the following for guidance.

    The training of an API model is executed in the ABS artifical intelligence (AI) engine.
    - Managing AI engine training
    - API reports using Postman
    - Accessing the PingIntelligence Dashboard

Copyright © 2025 Ping Identity Corporation

Next steps

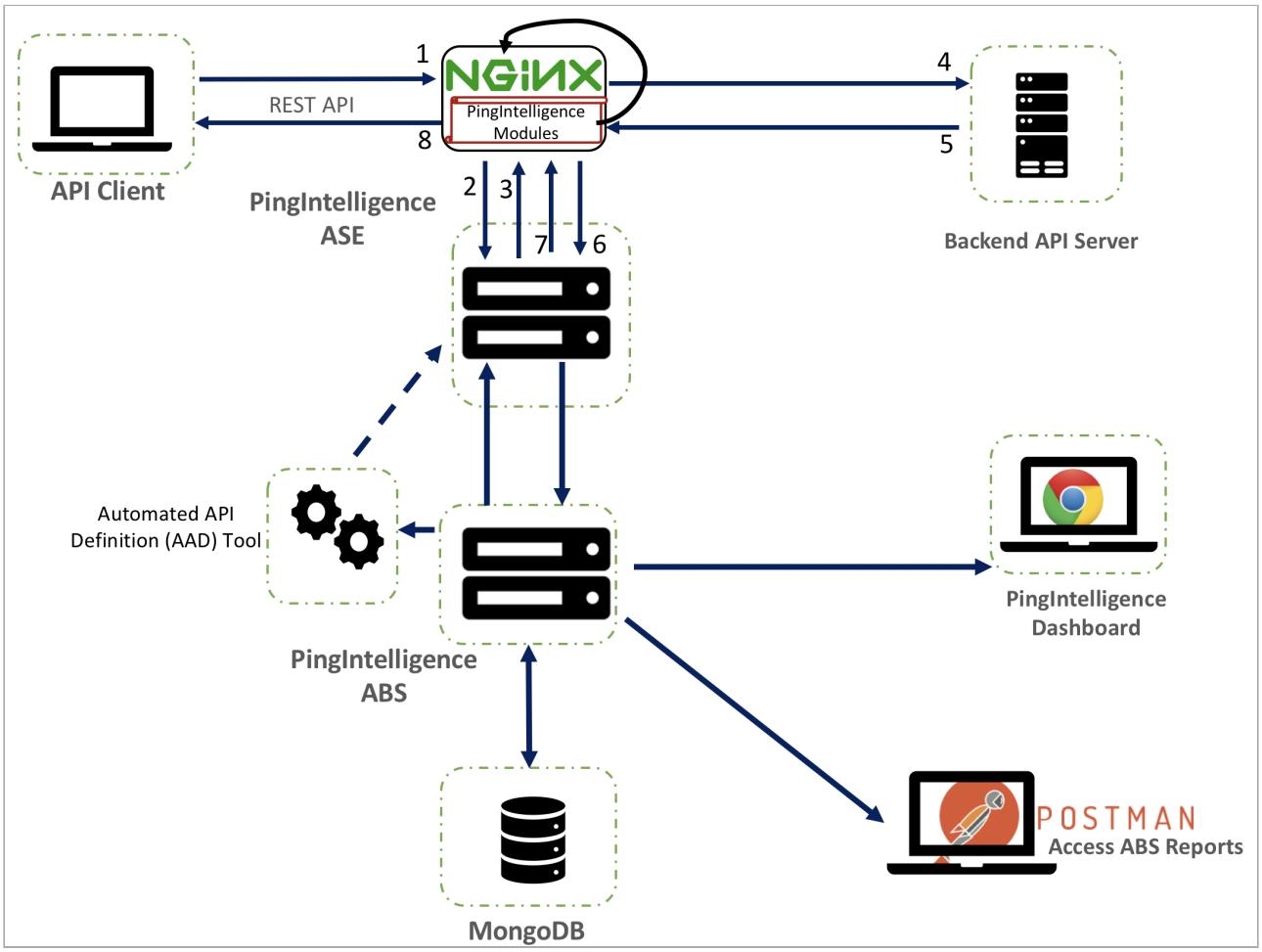
PingIntelligence API discovery is a process to discover and report APIs from your API environment. The discovered APIs are reported in the PingIntelligence Dashboard. APIs are discovered when a global API JavaScript Object Notation (JSON) is defined in the ASE.

For more information, see API discovery and configuration. You can edit the discovered API's JSON definition in the Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see Discovered APIs.

NGINX sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with NGINX.

The PingIntelligence policy modules are installed in the NGINX and pass API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.



The following is the traffic flow through NGINX and PingIntelligence for APIs components:

1. The client sends an incoming request to NGINX.
2. NGINX makes an API call to send the request metadata to ASE.

Copyright © 2025 Ping Identity Corporation

3. ASE checks the request against a registered set of APIs and looks for the origin IP, cookie, OAuth2 token, or API key in PingIntelligence AI engine-generated deny list. If all checks pass, ASE returns a `200-OK` response to the NGINX. If not, a different response code is sent to NGINX. The request information is also logged by ASE and sent to the AI engine for processing.
4. If NGINX receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, NGINX optionally blocks the client.
5. The response from the backend server is received by NGINX.
6. NGINX makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to NGINX.
8. NGINX sends the response received from the backend server to the client.

Preparing to deploy the PingIntelligence policy

Complete the following prerequisites before deploying the PingIntelligence policy.

About this task

The PingIntelligence policy modules are complied for NGINX 1.14.2. If you have a different version of NGINX, contact Ping Identity support.

Before deploying the PingIntelligence policy:

Steps

1. Install and configure the PingIntelligence software as follows.

   For more information on installing PingIntelligence, see PingIntelligence automated deployment for virtual machines and servers or PingIntelligence manual deployment.

   1. Sign on to your ASE machine and check that ASE is in `sideband` mode by running the following `status` command:

      ```
      /opt/pingidentity/ase/bin/cli.sh status
      ```

      Result:

Copyright © 2025 Ping Identity Corporation

```
API Security Enforcer
status                  : started
 mode : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : enabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

1. For secure communication between NGINX and ASE, enable sideband authentication by entering the following ASE command:

   ```
   # ./bin/cli.sh enable_sideband_authentication -u admin –p
   ```

2. To generate the sideband authentication token in ASE, enter the following command in the ASE command line and save the token for further use in Configuring NGINX for PingIntelligence.

   A token is required for NGINX to authenticate with ASE.

   ```
   # ./bin/cli.sh -u admin -p admin create_sideband_token
   ```

2. Configure the following for your operating system:

   Choose from:

   - RHEL 7.6:

     > **Important**
     >
     > The PingIntelligence modules for NGINX 1.14.2 are specifically compiled for RHEL 7.6 and OpenSSL `1.0.2k-fips` . If you do not have these specific versions of RHEL and OpenSSL, contact Ping Identity support.

     1. Verify your RHEL version by entering the following command on your machine:

        ```
        $ cat /etc/redhat-release
        Red Hat Enterprise Linux Server release 7.6 (Maipo)
        ```

     2. Open OpenSSL `1.0.2k-fips` on your RHEL 7.6 machine and check the OpenSSL version using the `openssl version` command:

Copyright © 2025 Ping Identity Corporation

```
$ openssl version
OpenSSL 1.0.2k-fips  26 Jan 2017
```

3. Extract the ASE certificate:

    1. Make sure that ASE is running. If ASE is not running, run the following command on ASE command line to start ASE:

    ```
    /opt/pingidentity/ase/bin/start.sh
    Starting API Security Enforcer 4.0.2...
    please see /opt/pingidentity/ase/logs/controller.log for more details
    ```

For more information on starting ASE, see Starting and stopping ASE

1. Run the following command to extract the ASE certificate and append it in the `test.ase.pi` file:

```
openssl s_client -connect  <ASE_IP>:<ASE_PORT>   2>/dev/null </dev/null |  sed -ne
'/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > test.ase.pi
```

2. Copy the certificate file to the NGINX machine and configure the certificate path in the `nginx.conf` file.

    1. Run the following command to download RHEL dependencies for compiling NGINX:

    ```
    # yum install pcre-devel.x86_64 openssl-devel.x86_64 zlib-devel.x86_64 wget
    gcc
    ```

- Ubuntu 16.0.4 LTS:

> **Important**
>
> The PingIntelligence modules are specifically compiled for Ubuntu 16.0.4 and OpenSSL `1.0.2g`. If you do not have these specific versions of Ubuntu and OpenSSL, contact Ping Identity support.

1. Run the following command to check your Ubuntu version:

```
$ cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04.6 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.6 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
```

Copyright © 2025 Ping Identity Corporation

2. OpenSSL `1.0.2g` and check the OpenSSL version using the `openssl version` command:

```
$ openssl version
OpenSSL 1.0.2g  26 Jan 2017
```

3. Extract the ASE certificate:

    1. Make sure that ASE is running. If ASE is not running, run the following command on ASE command line to start ASE:

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 4.0.2...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

For more information on starting ASE, see Starting and stopping ASE.

1. Run the following command to extract the ASE certificate and append in the `test.ase.pi` file:

```
openssl s_client -connect  <ASE_IP>:<ASE_PORT>   2>/dev/null </dev/null |  sed -ne '/-BEGIN
CERTIFICATE-/,/-END CERTIFICATE-/p' > test.ase.pi
```

2. Copy the certificate file to the NGINX machine and configure the certificate path in the `nginx.conf` file.

    1. Run the following command to download Ubuntu dependencies for compiling NGINX:

```
# apt-get -yq install make g++ gcc libpcre3 libpcre3-dev apt-utils zlib1g zlib1g-dev
curl openssl libssl-dev
```

Configuring NGINX

Configure NGINX for your operating system and then for PingIntelligence.

About this task

To configure NGINX, complete the following configuration steps:

1. Configure NGINX for either RHEL 7.6 or Ubuntu 16.04.
2. Configure NGINX for PingIntelligence.

Copyright © 2025 Ping Identity Corporation

RHEL 7.6

Configuring NGINX for RHEL 7.6

Configure NGINX for RHEL 7.6.

About this task

To compile NGINX Community Edition 1.14.2 for PingIntelligence for APIs:

Steps

1. Download the NGINX community version:

```
# wget https://nginx.org/download/nginx-1.14.2.tar.gz
```

2. Untar the NGINX file:

```
# tar -xvzf nginx-1.14.2.tar.gz
```

3. Change the directory to `nginx-1.14.2`:

```
# cd nginx-1.14.2
```

4. Compile and install NGINX by running the following command.

> **Note**
>
> These options for compiling NGINX are in addition to your environment specific options.

```
# ./configure --with-compat --with-http_ssl_module
```

   - `--with-compat`: This option enables NGINX to load dynamic modules.
   - `--with_http_ssl_module`: This flag is used configure SSL support in NGINX.

5. Run the `make` command to compile NGINX:

```
# make
```

6. Run the `make install` command to install NGINX:

```
# sudo make install
```

Copyright © 2025 Ping Identity Corporation

7. Verify the compilation by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -V
```

Result:

The output of the above command should display `--with-compat` and `--with_http_ssl_module` flags.

Copyright © 2025 Ping Identity Corporation

Ubuntu 16.04

Configuring NGINX for Ubuntu 16.04

Configure NGINX for Ubuntu 16.04.

About this task

To compile NGINX Community Edition 1.14.2 for PingIntelligence for APIs:

Steps

1. Download the NGINX community version:

```
# wget https://nginx.org/download/nginx-1.14.2.tar.gz
```

2. Untar the NGINX file:

```
# tar -xvzf nginx-1.14.2.tar.gz
```

3. Change the directory to `nginx-1.14.2`:

```
# cd nginx-1.14.2
```

4. Compile and install NGINX by running the following command.

> **Note**
>
> These options for compiling NGINX are in addition to your environment-specific options:
> - `--with-compat` : This option enables NGINX to load dynamic modules.
> - `--with_http_ssl_module` : This flag is used configure Secure Sockets Layer (SSL) support in NGINX.

```
# ./configure --with-compat --with-http_ssl_module
```

5. Run the `make` command to compile NGINX:

```
# make
```

6. Run the `make install` command to install NGINX:

```
# sudo make install
```

7. Verify the compilation by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -V
```

Copyright © 2025 Ping Identity Corporation

Result:

The output of the above command should display `--with-compat` and `--with_http_ssl_module` flags.

Configuring NGINX for PingIntelligence

Configure the `nginx.conf` setup NGINX and PingIntelligence sideband integration.

About this task

To configure NGINX for PingIntelligence:

Steps

1. Create a `modules` directory in NGINX:

```
# mkdir /usr/local/nginx/modules
```

2. Download the NGINX - PingIntelligence policy modules from the PingIntelligence Downloads site.
3. Untar the downloaded file.

```
tar -xvzf ubuntu_modules_1.14.2.tgz
modules/
modules/nginx-oss-list.txt
modules/ngx_ase_integration_module.so
modules/ngx_http_ase_integration_response_module.so
modules/ngx_http_ase_integration_request_module.so
```

4. Copy the three PingIntelligence modules for Ubuntu to the `modules` directory of NGINX.

   The three PingIntelligence modules are:

   - `ngx_ase_integration_module.so`
   - `ngx_http_ase_integration_request_module.so`
   - `ngx_http_ase_integration_response_module.so`

```
# cp ngx_ase_integration_module.so /usr/local/nginx/modules
# cp ngx_http_ase_integration_request_module.so /usr/local/nginx/modules
# cp ngx_http_ase_integration_response_module.so /usr/local/nginx/modules
```

5. Edit the `nginx.conf` file to load the PingIntelligence modules.

   Example:

   The following is a snippet of the `nginx.conf` file showing the loaded PingIntelligence modules. `IP:PORT` is the IP address of primary and secondary ASE.

Copyright © 2025 Ping Identity Corporation

```
worker_processes  1;

error_log  /usr/local/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /usr/local/nginx;

pid        /usr/local/nginx/pid/nginx.pid;

load_module modules/ngx_ase_integration_module.so; load_module modules/
ngx_http_ase_integration_request_module.so; load_module modules/
ngx_http_ase_integration_response_module.so;
events {
    worker_connections  1024;
}

http \{ keepalive_timeout 65; upstream pi.ase \{ server IP:PORT max_fails=1 max_conns=1024
fail_timeout=10; server IP:PORT max_fails=1 max_conns=1024 fail_timeout=10 backup; keepalive 32; }
 truncated nginx.conf
```

6. Add primary and secondary ASE hosts in `nginx.conf` in the upstream section.

   Example:

   The following is a snippet of the `nginx.conf` file with an ASE primary and secondary host configuration:

```
http {
    keepalive_timeout  65;
    upstream pi.ase {
        server 192.168.11.12:443 max_fails=3 max_conns=1024 fail_timeout=10; server
192.168.11.13:443 max_fails=3 max_conns=1024 fail_timeout=10 backup;
       keepalive 32;
   }
```

7. Configure an SSL certificate location and ASE sideband authentication token in `nginx.conf`. Copy the certificate to `/usr/local/nginx/ssl/test.ase.pi` on the NGINX machine and configure the certificate path in `nginx.conf` file.

   The ASE certificate was extracted from ASE in Preparing to deploy the PingIntelligence policy. The sideband authentication token was created in step 1c of Preparing to deploy the PingIntelligence policy.

   > **Note**
   >
   > You can also use your own SSL certificate by providing the path to the certificate in `set $certificate`. Make sure that ASE has the updated certificate.

   Example:

   The following is a snippet the showing certificate location and sideband authentication token:

Copyright © 2025 Ping Identity Corporation

```
#Certificiate location of ASE
     set $certificate /usr/local/nginx/ssl/test.ase.pi;
    #ASE Token for sideband authentication
     set $ase_token <YOUR ASE SIDEBAND TOKEN>;
```

8. Configure ASE request and response API endpoints in `nginx.conf`.

> **Note**
>
> `ase_integration_ssl_verify` is optional for non-SSL ASE connection.

Example:

The following snippet of `nginx.conf` shows ASE request and response:

```
                #ASE Request Proxy Configuration     location = /ase/request {
      internal;
      ase_integration https://pi.ase;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host pi.ase;
      ase_integration_ssl_trusted_certificate /usr/local/nginx/ssl/test.ase.pi;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name on;
      ase_integration_ssl_name test.ase.pi;
      ase_integration_next_upstream error timeout non_idempotent;

    #ASE Response Proxy Configuration     location = /ase/response {
      internal;
      ase_integration https://pi.ase;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host pi.ase;
      ase_integration_ssl_trusted_certificate /usr/local/nginx/ssl/test.ase.pi;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name on;
      ase_integration_ssl_name test.ase.pi;
      ase_integration_next_upstream error timeout non_idempotent;
```

9. Apply PingIntelligence modules for APIs by configuring `location` in `nginx.conf`.

`ase_integration_request` should be the first, and an `ase_integration_response` should be the last.

Copyright © 2025 Ping Identity Corporation

```
location /shop {
       ase_integration_request;
       proxy_pass http://localhost:8000/;
       ase_integration_response;
}
```

If you have more than more than one API, configure a `location` for each API as shown above.

10. Verify that `nginx.conf` is syntactically correct by running the following command:

```
# sudo /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

11. Restart NGINX by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -s stop
# sudo /usr/local/nginx/sbin/nginx
```

12. Run the following command to verify if `--with-compat` and `--with-http_ssl_module` is in the list of flags under configured arguments:

```
# sudo /usr/local/nginx/sbin/nginx -V
nginx version: nginx/1.14.2
built by gcc 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.11)
built with OpenSSL 1.0.2g  1 Mar 2016
TLS SNI support enabled
configure arguments: --with-compat --with-http_ssl_module
```

13. Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep 4443
```

Example:

The following is a sample `nginx.conf` for reference:

Copyright © 2025 Ping Identity Corporation

```
worker_processes 1; error_log /usr/local/nginx/logs/error.log debug; worker_rlimit_core 500M;
working_directory /usr/local/nginx; pid /usr/local/nginx/pid/nginx.pid; load_module modules/
ngx_ase_integration_module.so; load_module modules/ngx_http_ase_integration_request_module.so;
load_module modules/ngx_http_ase_integration_response_module.so;

events {
    worker_connections  1024;
}

http \{ keepalive_timeout 65; upstream pi.ase \{ server IP:PORT max_fails=1 max_conns=100
fail_timeout=10; server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup; keepalive 32; }

server {
    # remove "ssl" from the below line for a non-SSL frontend
    listen              4443 ssl bind;
    server_name         localhost;

    # Comment out the next 5-lines for a non-SSL frontend
    ssl_certificate     /usr/local/nginx/ssl/cert.pem;
    ssl_certificate_key /usr/local/nginx/ssl/key.pem;
    ssl_password_file   /usr/local/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    #root                /usr/share/nginx/html;
    #charset koi8-r;
    #access_log  /var/log/nginx/host.access.log  main;
    resolver 8.8.8.8 ipv6=off;

    #The following location configuration is to configure your application. A corresponding API JSON
should be present in ASE.
    location / {
       ase_integration_request;
       proxy_pass http://localhost:8080/;
       ase_integration_response;
       }
    #The following configuration is a Ping Intelligence configuration and do not edit
    set $correlationid $pid-$request_id-$server_addr-$remote_addr-$remote_port-$request_length-
$connection;

# ASE token must be configured
# ASE certificate must be copied under /usr/local/nginx/ssl/ and update the set $certificate to the
# certificate file path
#Certificate location of ASE
    set $certificate /usr/local/nginx/ssl/test.ase.pi;
    #ASE Token for sideband authentication
    set $ase_token <YOUR ASE SIDEBAND TOKEN HERE>;
    #Host header which should be send to ASE
    set $ase_host pi.ase;
    #SNI value to use for ASE
    set $ase_ssl_host pi.ase;
    #ASE Request Proxy Configuration
    location = /ase/request {
       internal;
```


Copyright © 2025 Ping Identity Corporation

```
        ase_integration https://pi.ase;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
    #ASE Response Proxy Configuration
    location = /ase/response {
        internal;
        ase_integration https://pi.ase;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
}
```

Integrating PingIntelligence

After the policy is deployed, integrate PingIntelligence.

Before you begin

Refer to the ASE and ABS AI Engine Admin Guides.

About this task

To integrate PingIntelligence:

Steps

1. Refer to the following:
    - ASE ports
    - API naming guidelines
    - Adding APIs to ASE in Defining an API using API JSON configuration file in sideband mode.

Copyright © 2025 Ping Identity Corporation

You can add individual APIs, or you can configure a global API. For more information on global API, see API discovery and configuration.

- ABS AI-based security

2. After you have added your APIs in ASE, train the API model. Refer to the following for guidance.

   The training of API model is completed in API Behavioral Security (ABS).

   - Training the ABS model
   - API reports using Postman.
   - Accessing the PingIntelligence Dashboard.

Next steps

PingIntelligence API discovery is a process to discover, and report APIs from your API environment. The discovered APIs are reported in the PingIntelligence Dashboard. APIs are discovered when a global API JavaScript Object Notation (JSON) is defined in the ASE.

For more information, see API discovery and configuration . You can edit the discovered API's JSON definition in the Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see Discovered APIs.

NGINX Plus sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with NGINX Plus.

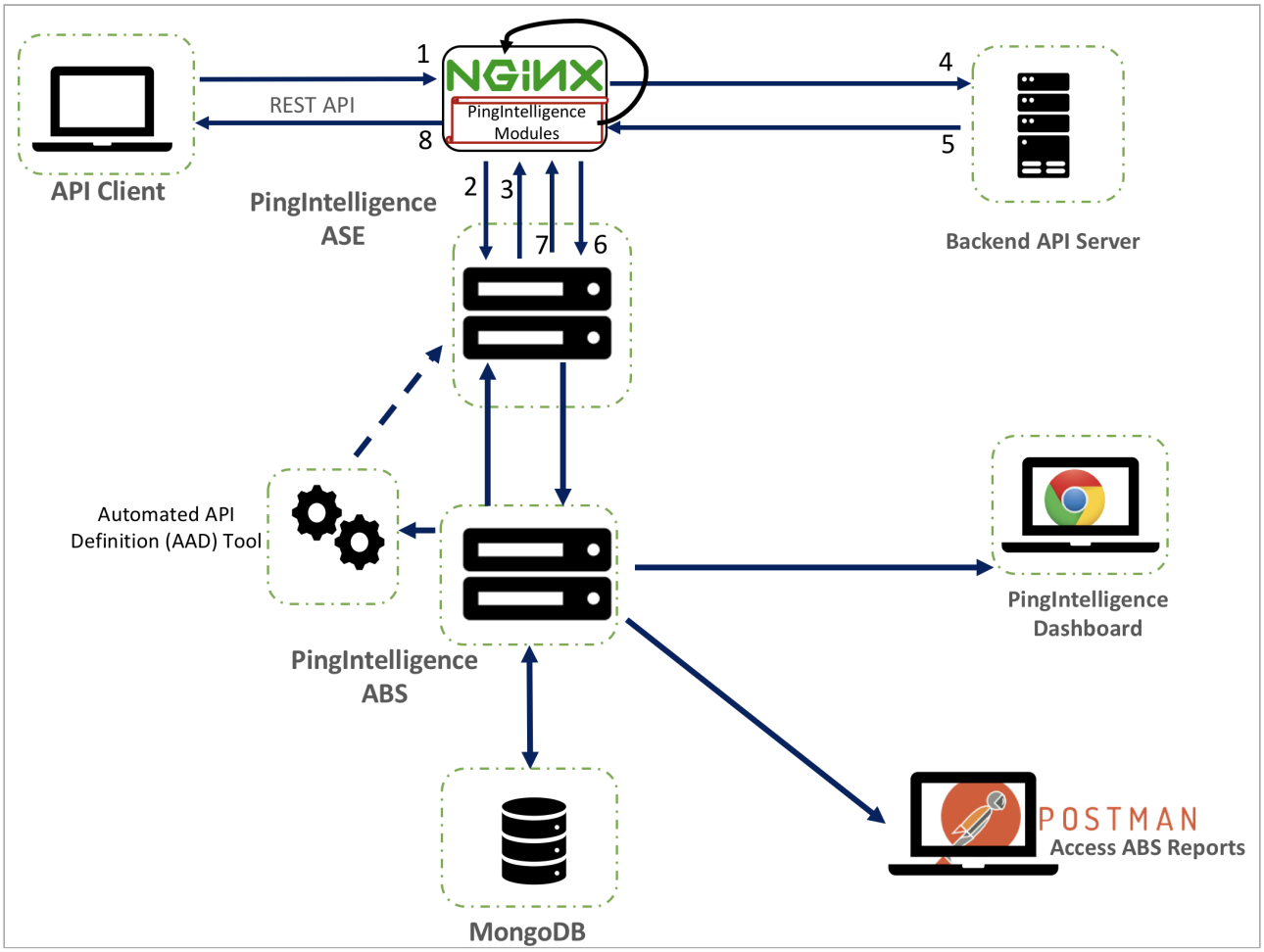
A PingIntelligence sideband policy is installed in NGINX Plus and passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking. PingIntelligence software adds support for reporting and attack detection based on usernames captured from token attributes.

Traffic flow for NGINX Plus integration with user information

Traffic flow for NGINX Plus integration without user information

Here is the traffic flow through NGINX and PingIntelligence for APIs components.

Copyright © 2025 Ping Identity Corporation

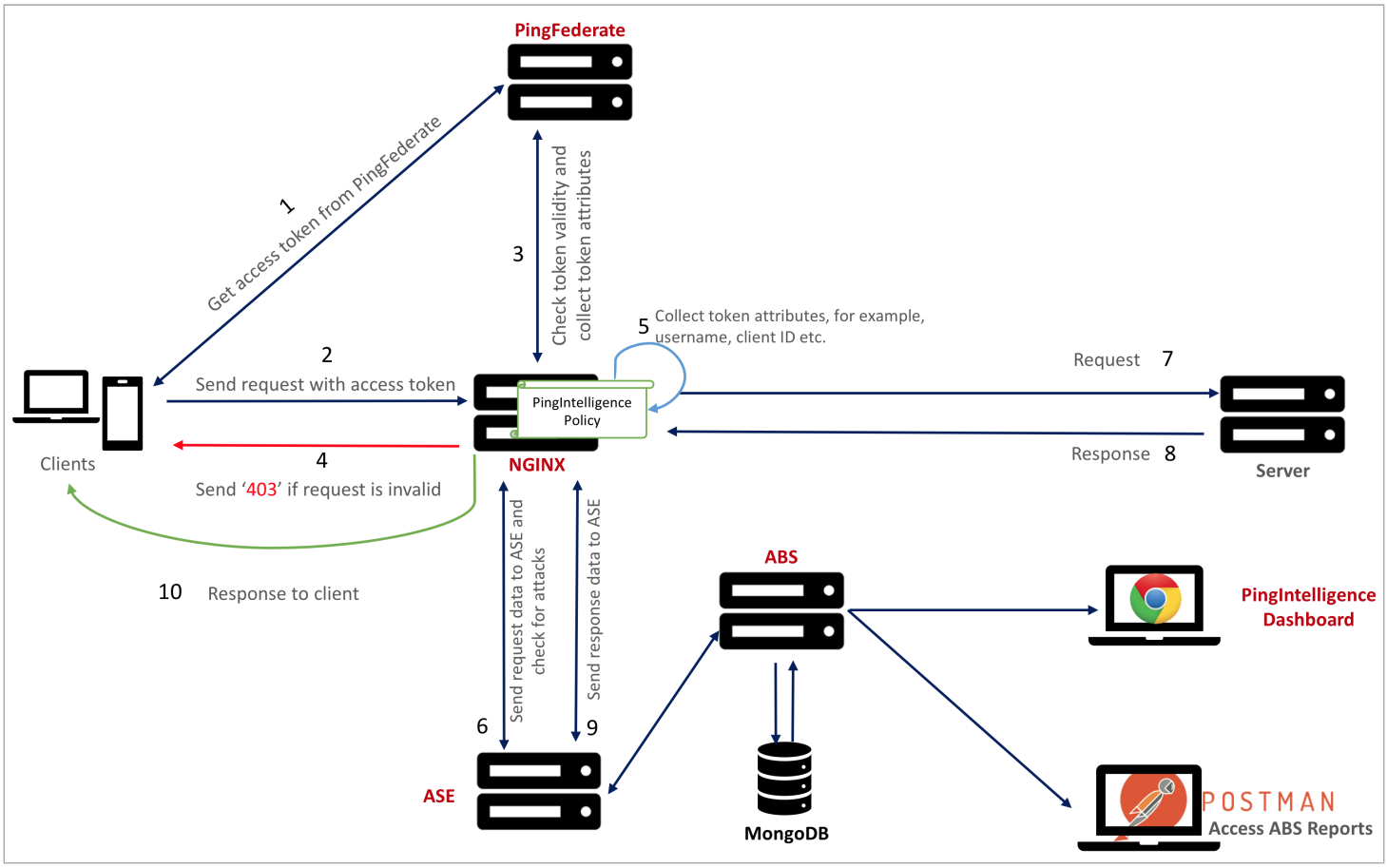


1. Client sends an incoming request to NGINX.
2. NGINX makes an API call to send the request metadata to ASE.
3. ASE checks the request against a registered set of APIs and looks for the origin IP, cookie, OAuth2 token, or API key in PingIntelligence AI engine generated Blacklist. If all checks pass, ASE returns a `200-OK` response to the NGINX. If not, a different response code is sent to NGINX. The request information is also logged by ASE and sent to the AI engine for processing.
4. If NGINX receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, NGINX optionally blocks the client.
5. The response from the backend server is received by NGINX.
6. NGINX makes a second API call to pass the response information to ASE which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a `200-OK` to NGINX.
8. NGINX sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

Traffic flow for NGINX Plus integration with user information

Below is the traffic flow through the NGINX Plus and PingIntelligence for APIs components. PingFederate is used as the OAuth server to gather the user information:



1. The client requests and receives an access token from PingFederate.
2. The client sends a request with the access token received from PingFederate.
3. NGINX Plus verifies the authenticity of the access token with PingFederate.
4. If the request is invalid, ASE sends a 403 error, and NGINX Plus drops the connection request.
5. If the token is valid, the PingIntelligence policy running in NGINX Plus collects API metadata and token attributes. In case of an invalid token, the request is allowed, however, without user information.
6. NGINX Plus makes an API call to send the request information to ASE. ASE checks the request against a registered set of APIs and checks the origin IP, cookie, or OAuth2 token against the AI-generated deny list. If all checks pass, ASE returns a `200-OK` response to the NGINX Plus. If not, a different response code is sent to NGINX Plus. The request information is also logged by ASE and sent to the AI engine for processing.
7. If NGINX Plus receives a `200-OK` response from ASE, then it forwards the request to the backend server. Otherwise, the Gateway optionally blocks the client.
8. The response from the backend server is received by NGINX Plus. NGINX Plus sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

9. NGINX Plus makes a second API call to pass the response information to ASE, which sends the information to the AI engine for processing. ASE receives the response information and sends a `200-OK` to NGINX Plus.
10. NGINX Plus sends the response to the client.

Preparing to deploy the PingIntelligence policy

Complete the following steps based on your operating system.

About this task

The PingIntelligence policy modules are complied for NGINX Plus R16. If you have a different version of NGINX Plus, contact Ping Identity support.

Before deploying the PingIntelligence policy:

Steps

1. Install and configure the PingIntelligence software.

   For more information, see PingIntelligence automated deployment for virtual machines and servers or PingIntelligence manual deployment.

2. Sign on to your ASE machine and verify that ASE is in `sideband` mode by running the following `status` command:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   ```

   Troubleshooting:

   If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

3. For secure communication between NGINX and ASE, enable sideband authentication by entering the following ASE command:

   ```
   # ./bin/cli.sh enable_sideband_authentication -u admin –p admin
   ```

4. To generate the token in ASE, enter the following command in the ASE command line and save the generated authentication token for further use.

   A token is required for NGINX to authenticate with ASE.

   ```
   # ./bin/cli.sh -u admin -p admin create_sideband_token
   ```

5. Configure the following for your operating system:

   Choose from:

   - RHEL 7.6

     1. Verify your RHEL version by entering the following command on your machine:

Copyright © 2025 Ping Identity Corporation

```
$ cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.6 (Maipo)
```

2. OpenSSL `1.0.2k-fips` on your RHEL 7.6 machine using the `openssl version` command:

```
$ openssl version
OpenSSL 1.0.2k-fips  26 Jan 2017
```

> **Important**
>
> The PingIntelligence modules for NGINX Plus have been specifically compiled for RHEL 7.6 and OpenSSL `1.0.2k-fips`. If you have different versions of these component, contact Ping Identity support.

3. Configure certificate for NGINX Plus:

    1. Create a directory for SSL certificates:

    ```
    # sudo mkdir -p /etc/ssl/nginx
    ```

    2. Sign on to the NGINX customer portal and download `nginx-repo.key` and `nginx-repo.crt` to `/etc/ss/nginx`.

For more information, see Installing NGINX Plus

1. Run the following command to download dependencies for RHEL:

```
# yum install wget ca-certificates
```

- Ubuntu 16.0.4 LTS

1. Run the following command to check your Ubuntu version:

```
$ cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.6 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
```

2. OpenSSL `1.0.2g` and check the OpenSSL version using the `openssl version` command:

Copyright © 2025 Ping Identity Corporation

```
$ openssl version
OpenSSL 1.0.2g  26 Jan 2017
```

3. Run the following command to download dependencies for Ubuntu:

```
# sudo apt-get install apt-transport-https lsb-release ca-certificates
```

4. Configure the certificate for NGINX Plus:

    1. Create a directory for SSL certificates:

    ```
    # sudo mkdir -p /etc/ssl/nginx
    ```

    2. Sign on to the NGINX customer portal and download `nginx-repo.key` and `nginx-repo.crt` to `/etc/ssl/nginx`.

    For more information, see Installing NGINX Plus.

    > **Important**
    >
    > The PingIntelligence modules are specifically compiled for Ubuntu 16.0.4 and OpenSSL `1.0.2g`. If you do not have these specific versions of Ubuntu and OpenSSL, contact Ping Identity support.

- Debian 9

1. Run the following command to check your Debian version:

```
$ cat /etc/os-release
 PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
VERSION_CODENAME=stretch
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

2. OpenSSL `1.1.0l` and check the OpenSSL version using the `openssl version` command:

```
$ openssl version
OpenSSL 1.1.0l  10 Sep 2019
```

3. Configure the certificate for NGINX Plus:

    1. Create a directory for SSL certificates:

Copyright © 2025 Ping Identity Corporation

```
# sudo mkdir -p /etc/ssl/nginx
```

2. Sign on to the NGINX customer portal and download `nginx-repo.key` and `nginx-repo.crt` to `/etc/ssl/nginx`.

For more information, see Installing NGINX Plus.

Installing NGINX Plus for RHEL 7.6

Install NGINX Plus for RHEL 7.6.

About this task

To install NGINX Plus:

Steps

1. Download NGINX Plus R16 repository:

```
# sudo wget -P /etc/yum.repos.d https://cs.nginx.com/static/files/nginx-plus-7.4.repo
```

2. Install the Lua modules:
    1. Check whether the Lua version `16+0.10.13-1.el7_4.ngx` is available in the list:

```
# sudo yum list nginx-plus-module-lua --showduplicates
```

    2. Install Lua module:

```
# sudo yum install nginx-plus-module-lua-16+0.10.13-1.el7_4.ngx
```

3. Install NGINX Plus:
    1. Check whether NGINX Plus version `nginx-plus-16-1.el7_4.ngx` is available in the list:

```
# sudo yum list nginx-plus --showduplicates
```

    2. Install NGINX Plus:

```
# sudo yum install nginx-plus-16-1.el7_4.ngx
```

Configuring NGINX Plus for PingIntelligence

Configure the `nginx.conf` to setup NGINX Plus and PingIntelligence sideband integration.

Copyright © 2025 Ping Identity Corporation

About this task

To configure NGINX Plus for PingIntelligence:

Steps

1. Create a `modules` directory in NGINX Plus:

   ```
   # mkdir /etc/nginx/modules
   ```

2. Download the NGINX Plus - PingIntelligence policy modules from the API Intelligence Downloads site.

3. Untar the downloaded file:

   ```
   # tar -xvzf pi-api-nginx-plus-policy-4.3.tar.gz
   ```

4. Copy the three PingIntelligence modules files for RHEL to the `modules` directory of NGINX Plus and `pi-pf.conf` file to `/usr/local/nginx/conf/` directory.

   The `pi-pf.conf` file has the OAuth policy details. The three PingIntelligence modules are:

   - `ngx_ase_integration_module.so`
   - `ngx_http_ase_integration_request_module.so`
   - `ngx_http_ase_integration_response_module.so`

   ```
   # cp ngx_ase_integration_module.so /etc/nginx/modules
   # cp ngx_http_ase_integration_request_module.so /etc/nginx/modules
   # cp ngx_http_ase_integration_response_module.so /etc/nginx/modules
   # cp pi-pf.conf /usr/local/nginx/conf/
   ```

5. Change to `root` user:

   ```
   # sudo su
   ```

6. Export client credentials as environment variables.

   Here `PF_ID` and `PF_SECRET` are PingFederate client ID and secret.

   ```
   # export PF_ID=<ID>
   # export PF_SECRET=<SECRET>
   ```

7. Edit the `nginx.conf` file to load the PingIntelligence modules.

   Example:

   The following is a snippet of the `nginx.conf` file showing the loaded PingIntelligence modules:

Copyright © 2025 Ping Identity Corporation

```
worker_processes  4;

error_log  /usr/local/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /usr/local/nginx;

pid        /usr/local/nginx/pid/nginx.pid;
env PF_ID;
env PF_SECRET;

load_module modules/ngx_ase_integration_module.so; load_module modules/
ngx_http_ase_integration_request_module.so; load_module modules/
ngx_http_ase_integration_response_module.so; load_module modules/ndk_http_module.so; load_module
modules/ngx_http_lua_module.so;

events {
    worker_connections  1024;
}
 truncated nginx.conf file
```

8. Configure ASE primary and secondary node IP address by replacing `IP:PORT` in the `nginx.conf` file snippet below:

```
http {

    keepalive_timeout  65;
    upstream ase.pi {
       server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10;
       server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
       #keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

truncated nginx.conf file
```

9. Configure introspect server IP address by replacing `IP:PORT` in the `nginx.conf` file snippet show below:

```
upstream introspect_server {
        server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10;
        server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10 backup;
        keepalive 32;
    }

truncated nginx.conf file
```

10. Configure the username and client ID keys in `nginx.conf`.

    These are the keys for username and client ID that you have configured in PingFederate.

Copyright © 2025 Ping Identity Corporation

```
set $oauth_username_key  Username;
set $oauth_client_id_key  ClientID;

truncated nginx.conf file
```

11. Configure the token parameter name after `$arg_` and in `ase/request`:

```
# Set the token parameter name below after $arg_ and inside /ase/request.
    set $oauth_key_param $arg_access_token;
    set $oauth_token_param $arg_access_token;

        #ASE Request Proxy Configuration
       location = /ase/request {
       internal;
       ase_integration https://test.ase.pi;
       ase_integration_method "POST";
       ase_integration_http_version 1.1;
       ase_integration_ase_token $ase_token;
       ase_integration_correlation_id $correlationid;
       ase_integration_host $ase_host;
       # set token key here.
       ase_integration_token_key  access_token;
       ase_integration_ssl_trusted_certificate $certificate;
       ase_integration_ssl_verify    off;
       ase_integration_ssl_verify_depth 1;
       ase_integration_ssl_server_name off;
       ase_integration_ssl_name $ase_ssl_host;
       ase_integration_next_upstream error timeout non_idempotent;
    }

truncated nginx.conf file
```

12. Configure the URL of the introspection server:

```
# Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

truncated nginx.conf file
```

13. Configure the ASE sideband token:

    The sideband authentication token was created in step 4 in Preparing to deploy the PingIntelligence policy.

    Example:

    The following is a snippet showing sideband authentication token:

```
 #ASE Token for sideband authentication
    set $ase_token <ASE_TOKEN>;
```

Copyright © 2025 Ping Identity Corporation

14. Configure the ASE request and response API endpoints in `nginx.conf`.

    The following snippet from `nginx.conf` shows ASE request and response:

```
#ASE Request Proxy Configuration
   location = /ase/request {
      internal;
      ase_integration https://test.ase.pi;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host $ase_host;
      # set token key here.
      ase_integration_token_key access_token;
      ase_integration_ssl_trusted_certificate $certificate;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name off;
      ase_integration_ssl_name $ase_ssl_host;
      ase_integration_next_upstream error timeout non_idempotent;
   }
   #ASE Response Proxy Configuration
   location = /ase/response {
      internal;
      ase_integration https://test.ase.pi;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host $ase_host;
      ase_integration_ssl_trusted_certificate $certificate;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name off;
      ase_integration_ssl_name $ase_ssl_host;
      ase_integration_next_upstream error timeout non_idempotent;
   }

truncated nginx.conf file
```

15. Apply the PingIntelligence policy at the global level (for all the APIs in your environment) or for an individual API.

> **Note**
>
> If the authorization header in the request has multiple tokens, the PingIntelligence policy extracts only the first valid bearer token from the authorization header.

Choose from:

- To apply the PingIntelligence policy globally, add `ase_integration_request` and `ase_integration_response` in the `server` section of `nginx.conf`:

Copyright © 2025 Ping Identity Corporation

```
server {
    listen              4443 ssl bind;
    server_name         localhost;
    ssl_certificate     /usr/local/nginx/ssl/cert.pem;
    ssl_certificate_key /usr/local/nginx/ssl/key.pem;
    ssl_password_file   /usr/local/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    resolver 8.8.8.8 ipv6=off;
     ase_integration_request; ase_integration_response;

    # Set OAuth Client details
 truncated nginx.conf file
```

- To apply the PingIntelligence policy for a specific API, configure `location` in `nginx.conf`.

  `ase_integration_request` should be the first, and an `ase_integration_response` should be the last.

> **Note**
>
> Comment-out `ase_integration_request` and `ase_integration_response` that was configured to apply PingIntelligence policy globally.

```
location / {
       include /usr/local/nginx/conf/pi-pf.conf;
       ase_integration_request;
       proxy_pass http://localhost:8080/;
       ase_integration_response;
}

truncated nginx.conf file
```

16. Verify the syntactical correctness of `nginx.conf` by running the following command:

```
# /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

17. Configure the PingIntelligence policy for NGINX Plus:

    1. Restart NGINX by entering the following command:

```
# /usr/local/nginx/sbin/nginx -s stop
# /usr/local/nginx/sbin/nginx
```

    2. Run the following command to verify if `--with-compat` and `--with-http_ssl_module` is in the list of flags under configured arguments:

Copyright © 2025 Ping Identity Corporation

```
# sudo /usr/local/nginx/sbin/nginx -V
nginx version: nginx/1.15.2 (nginx-plus-r16)
built by gcc 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.9)
built with OpenSSL 1.0.2g  1 Mar 2016
TLS SNI support enabled
configure arguments: --with-compat --with-http_ssl_module
```

3. Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep 4443
```

Example:

The following is a sample `nginx.conf` file. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file.

Copyright © 2025 Ping Identity Corporation

```
worker_processes  4;

error_log  /usr/local/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /usr/local/nginx;

pid        /usr/local/nginx/pid/nginx.pid;
env PF_ID;
env PF_SECRET;

load_module modules/ngx_ase_integration_module.so;
load_module modules/ngx_http_ase_integration_request_module.so;
load_module modules/ngx_http_ase_integration_response_module.so;
load_module modules/ndk_http_module.so;
load_module modules/ngx_http_lua_module.so;

events {
    worker_connections  1024;
}

http {

    keepalive_timeout  65;
    upstream test.ase.pi {
       server IP:PORT max_fails=1 max_conns=100 fail_timeout=10;
       server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
#      keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

    upstream introspect_server {
        server IP:PORT max_fails=1 max_conns=100 fail_timeout=10;
        server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup;
        keepalive 32;
    }

    lua_shared_dict cache_dict 128m;

server {
    listen              4443 ssl bind;
    server_name         localhost;
    ssl_certificate     /usr/local/nginx/ssl/cert.pem;
    ssl_certificate_key /usr/local/nginx/ssl/key.pem;
    ssl_password_file   /usr/local/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    resolver 8.8.8.8 ipv6=off;
    ase_integration_request;
    ase_integration_response;

    # Set OAuth Client details

    # Set env variable PF_ID &PF_SECRET
    set_by_lua $client_id 'return os.getenv("PF_ID")';
```

Copyright © 2025 Ping Identity Corporation

```
    set_by_lua $client_secret 'return os.getenv("PF_SECRET")';

    # Uncomment next 2 lines to set client credentials here.
    # set $client_id nginx_client;
    # set $client_secret nginx_secret;

    set $oauth_username_key Username;
    set $oauth_client_id_key ClientID;

    # Set the token parameter name below after $arg_ and inside /ase/request.
    set $oauth_key_param $arg_access_token;
    set $oauth_token_param $arg_access_token;
    # Set cache lifetime, default is 120s.
    set $oauth_cache_timeout 120;

    # Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

    location /introspect {
        internal;
        proxy_method    POST;
        if ($arg_auth_token) {
            set $auth_token $arg_auth_token;
        }
         if ($http_authorization ~* .*?(bearer)(\s+)([-a-zA-Z0-9._~+/]+)(,|\s|$)) {
            set $auth_token $3;
        }
        proxy_set_header  Content-Type "application/x-www-form-urlencoded";
        proxy_set_body  "client_id=${client_id}&client_secret=${client_secret}&token=$
{auth_token}";
        proxy_pass_request_body off;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_pass      $oauth_url;
    }

    location /shop {
       include /usr/local/nginx/conf/pi-pf.conf;
       proxy_pass http://18.209.173.37:4100/shop;

    }
    #DO NOT EDIT BELOW VARIABLE
    set $correlationid $pid-$request_id-$server_addr-$remote_addr-$remote_port-
$request_length-$connection;
    #Certificate location of ASE
    set $certificate /usr/local/nginx/ssl/test.ase.pi;
    #ASE Token for sideband authentication
    set $ase_token <ASE_TOKEN>;
    #Host header which should be send to ASE
    set $ase_host test.ase.pi;
    #SNI value to use for ASE
    set $ase_ssl_host test.ase.pi;
    #ASE Request Proxy Configuration
    location = /ase/request {
       internal;
```


Copyright © 2025 Ping Identity Corporation

```
        ase_integration https://test.ase.pi;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        # set token key here.
        ase_integration_token_key access_token;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
    #ASE Response Proxy Configuration
    location = /ase/response {
        internal;
        ase_integration https://test.ase.pi;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
}

}
```

Configuring NGINX Plus with PingAccess agent for PingIntelligence

You can install PingIntelligence sideband policy on NGINX Plus R22 or R23 systems with PingAccess agent.

Before you begin

Make sure the following prerequisites are complete before you configure NGINX Plus with PingIntelligence policy:

- API Security Enforcer (ASE) is installed, and the pre-conditions listed under prequisites for PingIntelligence are met.
- PingAccess and PingFederate are installed.
- PingAccess agent is installed and configured on NGINX. For more information, see PingAccess Agent for NGINX.
- PingAccess is configured to use PingFederate as a token provider and token introspection is enabled on PingAccess. For more information, see Configure PingFederate as the token provider for PingAccess.

About this task

Copyright © 2025 Ping Identity Corporation

Configure the `nginx.conf` to setup NGINX Plus and PingIntelligence sideband policy. Complete the following steps to integrate the sideband policy:

Steps

1. Download the NGINX Plus - PingIntelligence modules from the download site
2. Untar the downloaded file.

```
# tar -xvzf pi-api-nginx-plus-policy-5.0.tar
```

3. Copy the PingIntelligence modules files for RHEL to the modules directory of NGINX Plus and `pi-pf.conf` file to `/nginx/conf/` directory.
4. Change to `root` user.

```
# sudo su
```

5. Configure the `nginx.conf` file. Complete the following steps to configure `nginx.conf` for PingIntelligence:
   - Edit the `nginx.conf` file to load the PingIntelligence modules. Following is a snippet of `nginx.conf` file showing the loaded PingIntelligence module.

```
user  nginx;
worker_processes  auto;
error_log  /var/log/nginx/error.log debug;
pid        /var/run/nginx.pid;

load_module modules/ngx_ase_integration_module.so; load_module modules/
ngx_http_ase_integration_request_module.so; load_module modules/
ngx_http_ase_integration_response_module.so;load_module modules/ngx_http_paa_module.so;

events {
    worker_connections  1024;
}
```

> **Note**
>
> Make sure the modules are loaded in the order highlighted above.

   - Configure ASE primary and secondary node IP address by replacing IP:PORT in the `nginx.conf` file as shown in the following snippet.

Copyright © 2025 Ping Identity Corporation

```
http {

    upstream test.ase.pi {
       server  IP:PORT   max_fails=1 max_conns=100 fail_timeout=10;
       server  IP:PORT   max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
#      keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }
```

- Configure the following ASE parameters in `ngnix.conf` file:

Parameter	Description
`certificate`	Certificate location of ASE
`ase_token`	ASE Token for sideband authentication
`ase_host`	Host header that should be send to ASE
`ase_ssl_host`	SNI value to use for ASE

```
#DO NOT EDIT BELOW VARIABLE
    set $correlationid $pid-$request_id-$server_addr-$remote_addr-$remote_port-
$request_length-$connection;
    #Certificate location of ASE
    set  $certificate  /etc/ssl/nginx/test.ase.pi;
    #ASE Token for sideband authentication
    set  $ase_token  76748f33353940efab31e9fbe15d930a;
    #Host header which should be send to ASE
    set  $ase_host  test.ase.pi;
    #SNI value to use for ASE
    set  $ase_ssl_host  test.ase.pi;
```

6. Add PingIntelligence sideband policy

- To apply PingIntelligence policy globally, add `ase_integration_request` and `ase_integration_response` in the server section of `nginx.conf` as shown in the following snippet:

Copyright © 2025 Ping Identity Corporation

```
server {
    listen              44444 ssl bind;
    server_name         localhost;
    ssl_certificate     /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    add_header Allow "GET, POST, HEAD" always;
     ase_integration_request;
     ase_integration_response;

        truncated nginx.conf file
```

- To apply PingIntelligence sideband policy for a specific API, configure `location` in `nginx.conf` as shown in the following snippet:

```
location / {
       include /usr/local/nginx/conf/pi-pf.conf;
       ase_integration_request;
       proxy_pass http://localhost:8080/;
       ase_integration_response;
}

truncated nginx.conf file
```

Note: When configuring the policy for individual APIs, comment-out `ase_integration_request` and `ase_integration_response` that are added to apply PingIntelligence policy globally.

7. Run the following command and verify syntactical correctness of `nginx.conf` file:

```
# /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

8. Restart NGINX by entering the following command.

```
# /usr/local/nginx/sbin/nginx -s stop
# /usr/local/nginx/sbin/nginx
```

Next steps

- Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep <NGINX port number>
For example : # netstat -tulpn | grep 4443
```

- Configure API JSON file as explained in Configuring API JSON to extract user information.

Copyright © 2025 Ping Identity Corporation

Sample nginx.conf file - The following is a sample `nginx.conf` file.

Copyright © 2025 Ping Identity Corporation

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log debug;
pid        /var/run/nginx.pid;

load_module modules/ngx_ase_integration_module.so;
load_module modules/ngx_http_ase_integration_request_module.so;
load_module modules/ngx_http_ase_integration_response_module.so;
load_module modules/ngx_http_paa_module.so;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/paa/http.conf;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;
    upstream test.ase.pi {
       server 127.1.1.1:8443  max_fails=1 max_conns=100 fail_timeout=10;
       server 127.1.1.1:8443  max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
#      keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

  server {
    listen              44444 ssl bind;
    server_name         localhost;
    ssl_certificate     /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    add_header Allow "GET, POST, HEAD" always;
   location /todo/api-only {
       ase_integration_request;
       proxy_pass https://172.16.40.38:8443/todo/api-only;
       proxy_ssl_verify              off;
       ase_integration_response;
          }
    location /shopapi {
      proxy_pass https://172.16.40.70:4100/shopapi;
     proxy_ssl_verify              off;
          }
 #DO NOT EDIT BELOW VARIABLE
    set $correlationid $pid-$request_id-$server_addr-$remote_addr-$remote_port-$request_length-$connection;
```


Copyright © 2025 Ping Identity Corporation

```
    #Certificate location of ASE
    set $certificate /etc/ssl/nginx/test.ase.pi;
    #ASE Token for sideband authentication
    set $ase_token 76748f33353940efab31e9fbe15d930a;
    #Host header which should be send to ASE
    set $ase_host test.ase.pi;
    #SNI value to use for ASE
    set $ase_ssl_host test.ase.pi;
    #ASE Request Proxy Configuration
    location = /ase/request {
       internal;
       ase_integration https://test.ase.pi;
       ase_integration_method "POST";
       ase_integration_http_version 1.1;
       ase_integration_ase_token $ase_token;
       ase_integration_correlation_id $correlationid;
       ase_integration_host $ase_host;
       # set token key here.
       ase_integration_token_key access_token;
       ase_integration_ssl_trusted_certificate $certificate;
       ase_integration_ssl_verify    off;
       ase_integration_ssl_verify_depth 1;
       ase_integration_ssl_server_name off;
       ase_integration_ssl_name $ase_ssl_host;
       ase_integration_next_upstream error timeout non_idempotent;
    }
    #ASE Response Proxy Configuration
    location = /ase/response {
       internal;
       ase_integration https://test.ase.pi;
       ase_integration_method "POST";
       ase_integration_http_version 1.1;
       ase_integration_ase_token $ase_token;
       ase_integration_correlation_id $correlationid;
       ase_integration_host $ase_host;
       ase_integration_ssl_trusted_certificate $certificate;
       ase_integration_ssl_verify    off;
       ase_integration_ssl_verify_depth 1;
       ase_integration_ssl_server_name off;
       ase_integration_ssl_name $ase_ssl_host;
       ase_integration_next_upstream error timeout non_idempotent;
    }

location /introspect {
      internal;
        proxy_method    POST;
        if ($arg_auth_token) {
            set $auth_token $arg_auth_token;
        }
        if ($http_authorization ~* .?(bearer)(\s+)([-a-zA-Z0-9._~+/]+)(,|\s|$)) {
            set $auth_token $3;
        }
        #proxy_set_header  Content-Type "application/x-www-form-urlencoded";
        proxy_pass_request_body off;
        proxy_http_version 1.1;
```

Copyright © 2025 Ping Identity Corporation

```
        proxy_set_header Connection "";
        proxy_pass      $oauth_url;
        proxy_read_timeout   60;
        proxy_set_header authorization "";
    }

}

 include /etc/nginx/conf.d/.conf;

  }
```

Configuring API JSON to extract user information

This topic discusses the process to extract user attributes from signed JWT (JSON Web Token).

About this task

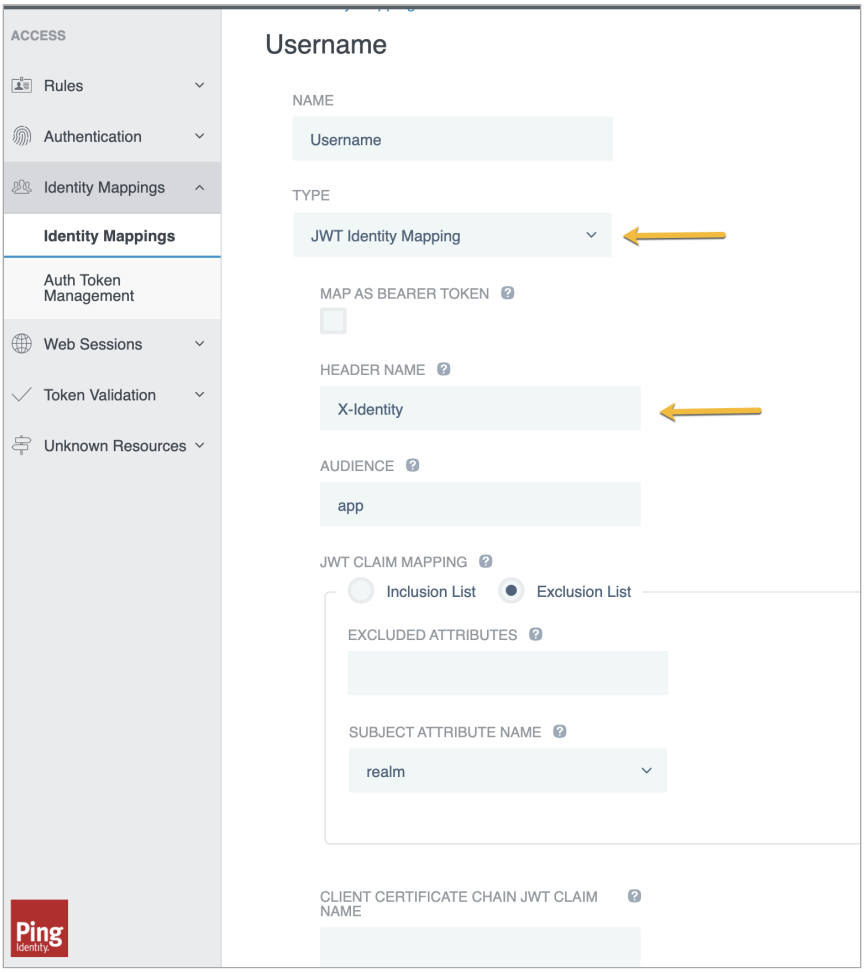
To extract the user information from the JSON Web Token (JWT) identity mapping:

Steps

1. Capture the value assigned to `HEADER NAME`.

   This is the name of the header to use when sending the signed JWT to the ASE.

Copyright © 2025 Ping Identity Corporation



2. Assign the value of `HEADER NAME` to the `location` key in the `jwt` object of the API JSON file.

   For more information, see Defining an API using API JSON configuration file in sideband mode.

```
"jwt": {
            "location": "h:X-Identity",
           "username": "username",
           "clientid": "client_id"
       }
```

Next steps

Refer to Creating a JWT identity mapping and Defining an API using API JSON configuration file in sideband mode.

Installing NGINX Plus for Ubuntu 16.0.4

Install NGINX Plus for Ubuntu 16.0.4.

About this task

To install NGINX Plus:

Copyright © 2025 Ping Identity Corporation

Steps

1. Download NGINX Plus R16 repository:

```
# printf "deb https://plus-pkgs.nginx.com/ubuntu lsb_release -cs nginx-plus\n" | sudo tee /etc/apt/
sources.list.d/nginx-plus.list
# sudo wget -q -O /etc/apt/apt.conf.d/90nginx https://cs.nginx.com/static/files/90nginx
# sudo apt-get update
```

2. Install NGINX Plus with Lua modules:

    1. Check whether `16-1~xenial` is available in the list:

    ```
    # sudo apt-cache show nginx-plus | grep "Version"
    ```

    2. Install NGINX Plus:

    ```
    # sudo apt-get install nginx-plus=16-1~xenial
    # sudo apt-get install nginx-plus-module-ndk=16+0.3.0-1~xenial
    # sudo apt-get install nginx-plus-module-lua=16+0.10.13-2~xenial
    ```

Configuring NGINX Plus for PingIntelligence

Configure `nginx.conf` to set up NGINX Plus and PingIntelligence sideband integration.

About this task

To configure NGINX Plus for PingIntelligence:

Steps

1. Create a `modules` directory in NGINX Plus:

```
# mkdir /etc/nginx/modules
```

2. Download the NGINX Plus - PingIntelligence policy modules from the PingIntelligence Downloads site.

3. Untar the downloaded file:

```
# tar -xvzf pi-api-nginx-plus-policy-4.3.tar.gz
```

4. Copy the three PingIntelligence modules files for Ubuntu to the `modules` directory of NGINX Plus and the `pi-pf.conf` file to `/usr/local/nginx/conf/` directory.

   The three PingIntelligence modules are:

   - `ngx_ase_integration_module.so`

Copyright © 2025 Ping Identity Corporation

- `ngx_http_ase_integration_request_module.so`
- `ngx_http_ase_integration_response_module.so`

The `pi-pf.conf` file has the OAuth policy details.

```
# cp ngx_ase_integration_module.so /etc/nginx/modules
# cp ngx_http_ase_integration_request_module.so /etc/nginx/modules
# cp ngx_http_ase_integration_response_module.so /etc/nginx/modules
# cp pi-pf.conf /usr/local/nginx/conf/
```

5. Change to `root` user:

```
# sudo su
```

6. Export client credentials as environment variables.

```
# export PF_ID=<ID>
# export PF_SECRET=<SECRET>
```

Here `PF_ID` and `PF_SECRET` are PingFederate client ID and secret.

7. Edit the `nginx.conf` file to load the PingIntelligence modules.

Example:

The following is a snippet of the `nginx.conf` file showing the loaded PingIntelligence modules:

```
worker_processes  4;

error_log  /usr/local/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /usr/local/nginx;

pid        /usr/local/nginx/pid/nginx.pid;
env PF_ID;
env PF_SECRET;

load_module modules/ngx_ase_integration_module.so; load_module modules/
ngx_http_ase_integration_request_module.so; load_module modules/
ngx_http_ase_integration_response_module.so; load_module modules/ndk_http_module.so; load_module
modules/ngx_http_lua_module.so;

events {
    worker_connections  1024;
}
 truncated nginx.conf file
```

8. Configure ASE primary and secondary node IP address by replacing `IP:PORT` in the `nginx.conf` file snippet shown below:

Copyright © 2025 Ping Identity Corporation

Example:

```
http {

    keepalive_timeout  65;
    upstream ase.pi {
       server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10;
       server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
       #keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

truncated nginx.conf file
```

9. Configure introspect server IP address by replacing `IP:PORT` in the `nginx.conf` file snippet show below:

   Example:

```
upstream introspect_server {
        server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10;
        server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10 backup;
        keepalive 32;
    }

truncated nginx.conf file
```

10. Configure the username and client ID keys in `nginx.conf`.

    These are the keys for username and client ID that you have configured in PingFederate.

```
set $oauth_username_key  Username;
set $oauth_client_id_key  ClientID;

truncated nginx.conf file
```

11. Configure the token parameter name after `$arg_` and in `ase/request`:

Copyright © 2025 Ping Identity Corporation

```
# Set the token parameter name below after $arg_ and inside /ase/request.
    set $oauth_key_param $arg_access_token;
    set $oauth_token_param $arg_access_token;
        #ASE Request Proxy Configuration
       location = /ase/request {
       internal;
       ase_integration https://test.ase.pi;
       ase_integration_method "POST";
       ase_integration_http_version 1.1;
       ase_integration_ase_token $ase_token;
       ase_integration_correlation_id $correlationid;
       ase_integration_host $ase_host;
       # set token key here.
       ase_integration_token_key  access_token;
       ase_integration_ssl_trusted_certificate $certificate;
       ase_integration_ssl_verify    off;
       ase_integration_ssl_verify_depth 1;
       ase_integration_ssl_server_name off;
       ase_integration_ssl_name $ase_ssl_host;
       ase_integration_next_upstream error timeout non_idempotent;
    }

truncated nginx.conf file
```

12. Configure the URL of the introspection server:

```
# Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

truncated nginx.conf file
```

13. Configure the ASE sideband token.

    The sideband authentication token was created in step 4 of Preparing to deploy the PingIntelligence policy.

    Example:

    The following is a snippet the showing certificate location and sideband authentication token:

```
 #ASE Token for sideband authentication
    set $ase_token  <ASE_TOKEN>;
```

14. Configure the ASE request and response API endpoints in `nginx.conf`.

    Example:

    The following snippet of `nginx.conf` shows ASE request and response:

Copyright © 2025 Ping Identity Corporation

```
#ASE Request Proxy Configuration
   location = /ase/request {
      internal;
      ase_integration https://test.ase.pi;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host $ase_host;
      # set token key here.
      ase_integration_token_key access_token;
      ase_integration_ssl_trusted_certificate $certificate;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name off;
      ase_integration_ssl_name $ase_ssl_host;
      ase_integration_next_upstream error timeout non_idempotent;
   }
   #ASE Response Proxy Configuration
   location = /ase/response {
      internal;
      ase_integration https://test.ase.pi;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host $ase_host;
      ase_integration_ssl_trusted_certificate $certificate;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name off;
      ase_integration_ssl_name $ase_ssl_host;
      ase_integration_next_upstream error timeout non_idempotent;
   }

truncated nginx.conf file
```

15. Apply the PingIntelligence policy either at the global level for all the APIs in your environment or for an individual API.

> **Note**
>
> If the authorization header in the request has multiple tokens, the PingIntelligence policy extracts only the first valid bearer token from the authorization header.

Choose from:

- To apply the PingIntelligence policy globally, add `ase_integration_request` and `ase_integration_response` in the `server` section of `nginx.conf` as shown below:

Copyright © 2025 Ping Identity Corporation

```
server {
    listen              4443 ssl bind;
    server_name         localhost;
    ssl_certificate     /usr/local/nginx/ssl/cert.pem;
    ssl_certificate_key /usr/local/nginx/ssl/key.pem;
    ssl_password_file   /usr/local/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    resolver 8.8.8.8 ipv6=off;
     ase_integration_request; ase_integration_response;

    # Set OAuth Client details
 truncated nginx.conf file
```

- To apply the PingIntelligence policy for a specific API, configure `location` in `nginx.conf`.

  `ase_integration_request` should be the first and a `ase_integration_response` should be the last.

  > **Note**
  >
  > Comment-out the `ase_integration_request` and `ase_integration_response` that was configured to apply the PingIntelligence policy globally.

```
location / {
       include /usr/local/nginx/conf/pi-pf.conf;
       ase_integration_request;
       proxy_pass http://localhost:8080/;
       ase_integration_response;
}

truncated nginx.conf file
```

16. To verify the syntactical correctness of `nginx.conf`, run the following command:

```
# /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

17. Configure the PingIntelligence policy for NGINX Plus:

    1. Restart NGINX by entering the following command:

```
# /usr/local/nginx/sbin/nginx -s stop
# /usr/local/nginx/sbin/nginx
```

    2. Run the following command to verify if `--with-compat` and `--with-http_ssl_module` is in the list of flags under configured arguments:

Copyright © 2025 Ping Identity Corporation

```
# sudo /usr/local/nginx/sbin/nginx -V
nginx version: nginx/1.15.2 (nginx-plus-r16)
built by gcc 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.9)
built with OpenSSL 1.0.2g  1 Mar 2016
TLS SNI support enabled
configure arguments: --with-compat --with-http_ssl_module
```

3. Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep 4443
```

Example:

The following is a sample `nginx.conf` file. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file.

Copyright © 2025 Ping Identity Corporation

```
worker_processes  4;

error_log  /usr/local/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /usr/local/nginx;

pid        /usr/local/nginx/pid/nginx.pid;
env PF_ID;
env PF_SECRET;

load_module modules/ngx_ase_integration_module.so;
load_module modules/ngx_http_ase_integration_request_module.so;
load_module modules/ngx_http_ase_integration_response_module.so;
load_module modules/ndk_http_module.so;
load_module modules/ngx_http_lua_module.so;

events {
    worker_connections  1024;
}

http {

    keepalive_timeout  65;
    upstream test.ase.pi {
       server IP:PORT max_fails=1 max_conns=100 fail_timeout=10;
       server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
#      keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

    upstream introspect_server {
        server IP:PORT max_fails=1 max_conns=100 fail_timeout=10;
        server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup;
        keepalive 32;
    }

    lua_shared_dict cache_dict 128m;

server {
    listen              4443 ssl bind;
    server_name         localhost;
    ssl_certificate     /usr/local/nginx/ssl/cert.pem;
    ssl_certificate_key /usr/local/nginx/ssl/key.pem;
    ssl_password_file   /usr/local/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    resolver 8.8.8.8 ipv6=off;
    ase_integration_request;
    ase_integration_response;

    # Set OAuth Client details

    # Set env variable PF_ID &PF_SECRET
    set_by_lua $client_id 'return os.getenv("PF_ID")';
```

Copyright © 2025 Ping Identity Corporation

```
    set_by_lua $client_secret 'return os.getenv("PF_SECRET")';

    # Uncomment next 2 lines to set client credentials here.
    # set $client_id nginx_client;
    # set $client_secret nginx_secret;

    set $oauth_username_key Username;
    set $oauth_client_id_key ClientID;

    # Set the token parameter name below after $arg_ and inside /ase/request.
    set $oauth_key_param $arg_access_token;
    set $oauth_token_param $arg_access_token;

    # Set cache lifetime, default is 120s.
    set $oauth_cache_timeout 120;

    # Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

    location /introspect {
        internal;
        proxy_method    POST;
        if ($arg_auth_token) {
            set $auth_token $arg_auth_token;
        }
         if ($http_authorization ~* .*?(bearer)(\s+)([-a-zA-Z0-9._~+/]+)(,|\s|$)) {
            set $auth_token $3;
        }
        proxy_set_header  Content-Type "application/x-www-form-urlencoded";
        proxy_set_body  "client_id=${client_id}&client_secret=${client_secret}&token=$
{auth_token}";
        proxy_pass_request_body off;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_pass      $oauth_url;
    }

    location /shop {
       include /usr/local/nginx/conf/pi-pf.conf;
       proxy_pass http://18.209.173.37:4100/shop;

    }
    #DO NOT EDIT BELOW VARIABLE
    set $correlationid $pid-$request_id-$server_addr-$remote_addr-$remote_port-
$request_length-$connection;
    #Certificate location of ASE
    set $certificate /usr/local/nginx/ssl/test.ase.pi;
    #ASE Token for sideband authentication
    set $ase_token <ASE_TOKEN>;
    #Host header which should be send to ASE
    set $ase_host test.ase.pi;
    #SNI value to use for ASE
    set $ase_ssl_host test.ase.pi;
    #ASE Request Proxy Configuration
    location = /ase/request {
```


Copyright © 2025 Ping Identity Corporation

```
        internal;
        ase_integration https://test.ase.pi;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        # set token key here.
        ase_integration_token_key access_token;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
    #ASE Response Proxy Configuration
    location = /ase/response {
        internal;
        ase_integration https://test.ase.pi;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
}

}
```

Installling NGINX Plus for Debian 9

Install NGINX Plus for Debian 9.

About this task

Tto install NGINX Plus:

Steps

1. Download NGINX Plus R19 repository:

```
# printf "deb https://plus-pkgs.nginx.com/debian lsb_release -cs nginx-plus\n" | sudo tee /etc/apt/
sources.list.d/nginx-plus.list
# sudo wget -q -O /etc/apt/apt.conf.d/90nginx https://cs.nginx.com/static/files/90nginx
# sudo apt-get update
```

Copyright © 2025 Ping Identity Corporation

2. Install NGINX Plus:

    1. Check whether `19-1~stretch` is available in the list:

    ```
    # sudo apt-cache show nginx-plus | grep "Version"
    ```

    2. Install NGINX Plus and related modules:

    ```
    # sudo apt-get install nginx-plus=19-1~stretch
    # sudo apt-get install nginx-plus-module-ndk=19+0.3.0-1~stretch
    # sudo apt-get install nginx-plus-module-lua=19+0.10.15-1~stretch
    ```

Configuring NGINX Plus for PingIntelligence

Complete the following steps to configure the `nginx.conf` file to set up NGINX Plus and PingIntelligence sideband integration.

About this task

To configure NGINX Plus for PingIntelligence:

Steps

1. Create a `modules` directory in NGINX Plus if it has not already been created:

    ```
    # mkdir /etc/nginx/modules
    ```

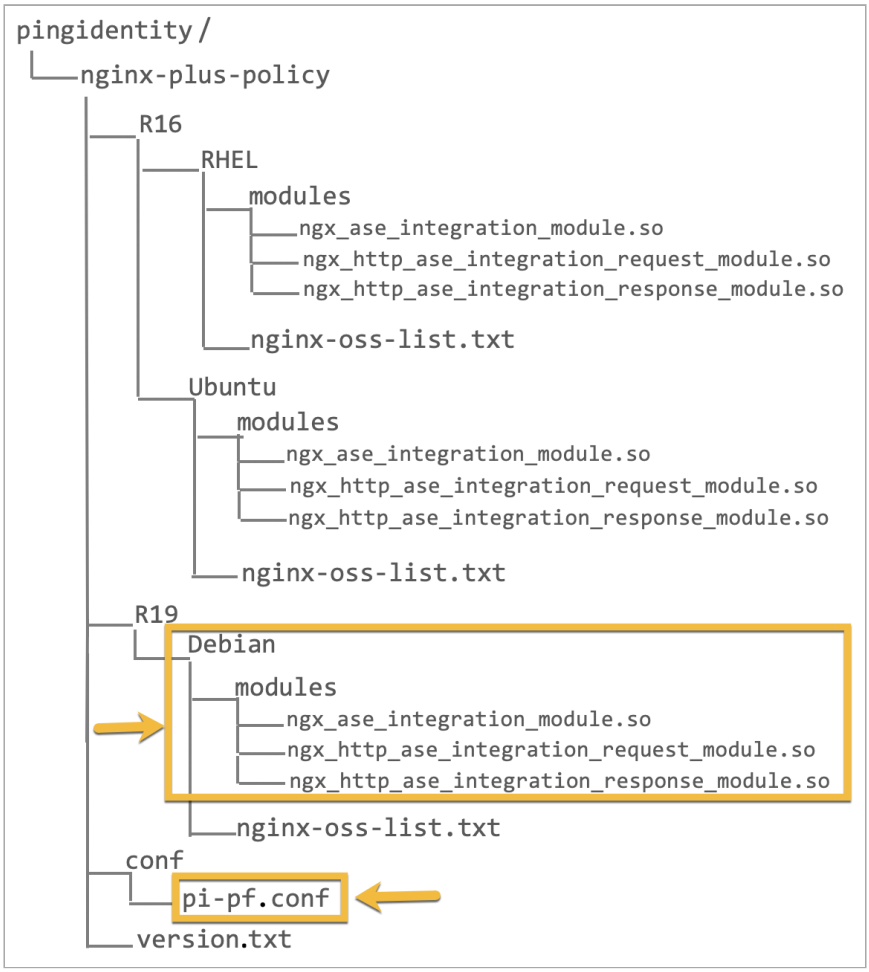
2. Download the NGINX Plus - PingIntelligence policy modules from the API Intelligence Downloads site.

3. Untar the downloaded file:

    ```
    # tar -xvzf pi-api-nginx-plus-policy-4.3.tar.gz
    ```

    The following is the directory structure.

Copyright © 2025 Ping Identity Corporation



4. Copy the three PingIntelligence modules files for Debian to the `/etc/nginx/modules/` directory of NGINX Plus and the `pi-pf.conf` file to the `/etc/nginx/` directory.

   The `pi-pf.conf` file has the OAuth policy details.

```
# cp ngx_ase_integration_module.so /etc/nginx/modules/
# cp ngx_http_ase_integration_request_module.so /etc/nginx/modules/
# cp ngx_http_ase_integration_response_module.so /etc/nginx/modules/
# cp pi-pf.conf /etc/nginx/
```

5. Change to `root` user:

```
# sudo su
```

6. Export client credentials as environment variables:

```
# export PF_ID=<ID>
# export PF_SECRET=<SECRET>
```

   Here `PF_ID` and `PF_SECRET` are PingFederate client ID and secret.

Copyright © 2025 Ping Identity Corporation

7. Edit the `nginx.conf` file to load the PingIntelligence modules.

   Example:

   The following is a snippet of the `nginx.conf` file showing the loaded PingIntelligence modules:

```
worker_processes  4;

error_log  /etc/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /etc/nginx;

pid        /etc/nginx/pid/nginx.pid;
env PF_ID;
env PF_SECRET;

load_module modules/ngx_ase_integration_module.so; load_module modules/
ngx_http_ase_integration_request_module.so; load_module modules/
ngx_http_ase_integration_response_module.so; load_module modules/ndk_http_module.so; load_module
modules/ngx_http_lua_module.so;

events {
    worker_connections  1024;
}
 truncated nginx.conf file
```

8. Configure ASE primary and secondary node IP address by replacing `IP:PORT` in the `nginx.conf` file snippet show below:

```
http {

    keepalive_timeout  65;
    upstream test.ase.pi {
       server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10;
       server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
       #keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

truncated nginx.conf file
```

9. Configure the introspect server IP address by replacing `IP:PORT` in the `nginx.conf` file snippet shown below.

   Example:

```
upstream introspect_server {
        server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10;
        server  IP:PORT  max_fails=1 max_conns=100 fail_timeout=10 backup;
        keepalive 32;
    }

truncated nginx.conf file
```


Copyright © 2025 Ping Identity Corporation

10. Configure the username and client ID keys in `nginx.conf`.

    These are the keys for username and client ID that you have configured in PingFederate.

```
set $oauth_username_key  Username;
set $oauth_client_id_key  ClientID;

truncated nginx.conf file
```

11. Configure the token parameter name after `$arg_` and in `ase/request`:

```
# Set the token parameter name below after $arg_ and inside /ase/request.
    set $oauth_key_param $arg_access_token;
    set $oauth_token_param $arg_access_token;
        #ASE Request Proxy Configuration
       location = /ase/request {
       internal;
       ase_integration https://test.ase.pi;
       ase_integration_method "POST";
       ase_integration_http_version 1.1;
       ase_integration_ase_token $ase_token;
       ase_integration_correlation_id $correlationid;
       ase_integration_host $ase_host;
       # set token key here.
       ase_integration_token_key  access_token;
       ase_integration_ssl_trusted_certificate $certificate;
       ase_integration_ssl_verify    off;
       ase_integration_ssl_verify_depth 1;
       ase_integration_ssl_server_name off;
       ase_integration_ssl_name $ase_ssl_host;
       ase_integration_next_upstream error timeout non_idempotent;
    }

truncated nginx.conf file
```

12. Configure the URL of the introspection server:

```
# Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

truncated nginx.conf file
```

13. Configure the ASE sideband token.

    The sideband authentication token was created in step 4 in Preparing to deploy the PingIntelligence policy. The snippet in step 14 shows the certificate location and sideband authentication token.

14. Configure the ASE request and response API endpoints in `nginx.conf`.

    Example:

Copyright © 2025 Ping Identity Corporation

The following snippet of `nginx.conf` shows the ASE request and response:

```
#ASE Request Proxy Configuration
   location = /ase/request {
      internal;
      ase_integration https://test.ase.pi;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host $ase_host;
      # set token key here.
      ase_integration_token_key access_token;
      ase_integration_ssl_trusted_certificate $certificate;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name off;
      ase_integration_ssl_name $ase_ssl_host;
      ase_integration_next_upstream error timeout non_idempotent;
   }
   #ASE Response Proxy Configuration
   location = /ase/response {
      internal;
      ase_integration https://test.ase.pi;
      ase_integration_method "POST";
      ase_integration_http_version 1.1;
      ase_integration_ase_token $ase_token;
      ase_integration_correlation_id $correlationid;
      ase_integration_host $ase_host;
      ase_integration_ssl_trusted_certificate $certificate;
      ase_integration_ssl_verify    off;
      ase_integration_ssl_verify_depth 1;
      ase_integration_ssl_server_name off;
      ase_integration_ssl_name $ase_ssl_host;
      ase_integration_next_upstream error timeout non_idempotent;
   }

truncated nginx.conf file
```

15. Apply the PingIntelligence policy either at the global level for all the APIs in your environment or for an individual API.

> **Note**
>
> If the authorization header in the request has multiple tokens, the PingIntelligence policy extracts only the first valid bearer token from the authorization header.

Choose from:

- To apply the PingIntelligence policy globally, add `ase_integration_request` and `ase_integration_response` in the `server` section of the `nginx.conf` file as shown below:

Copyright © 2025 Ping Identity Corporation

```
server {
    listen              4443 ssl bind;
    server_name         localhost;
    ssl_certificate     /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    ssl_password_file   /etc/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    resolver 8.8.8.8 ipv6=off;
     ase_integration_request; ase_integration_response;

    # Set OAuth Client details
 truncated nginx.conf file
```

- To apply PingIntelligence modules for specific APIs, configure `location` in `nginx.conf`.

  `ase_integration_request` should be the first and `ase_integration_response` should be the last, as shown in the following snippet.

  > **Note**
  >
  > When applying the policy to a specific API, comment-out `ase_integration_request` and `ase_integration_response`, which are configured in the server section of the `nginx.conf` file to apply the PingIntelligence policy globally.

```
location / {
       include /etc/nginx/pi-pf.conf;
       ase_integration_request;
       proxy_pass http://localhost:8080/;
       ase_integration_response;
}

truncated nginx.conf file
```

16. To verify the syntactical correctness of `nginx.conf`, run the following command:

```
# /usr/sbin/nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

17. Configure the PingIntelligence policy for NGINX Plus:

    1. Restart NGINX by entering the following command:

```
# /usr/sbin/nginx -s stop
# /usr/sbin/nginx
```

    2. Run the following command to verify if `--with-compat` and `--with-http_ssl_module` is in the list of flags under configured arguments:

Copyright © 2025 Ping Identity Corporation

```
# sudo /usr/sbin/nginx -V
nginx version: nginx/1.17.3 (nginx-plus-r19)
built by gcc 6.3.0 20170516 (Debian 6.3.0-18+deb9u1)
built with OpenSSL 1.1.0j 20 Nov 2018 (running with OpenSSL 1.1.0l 10 Sep 2019)
TLS SNI support enabled
```

3. Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep 4443
```

Example:

The following is a sample `nginx.conf` file. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file.

Copyright © 2025 Ping Identity Corporation

```
worker_processes  4;

error_log  /etc/nginx/logs/error.log debug;
worker_rlimit_core  500M;
working_directory  /etc/nginx;

pid        /etc/nginx/pid/nginx.pid;
env PF_ID;
env PF_SECRET;

load_module modules/ngx_ase_integration_module.so;
load_module modules/ngx_http_ase_integration_request_module.so;
load_module modules/ngx_http_ase_integration_response_module.so;
load_module modules/ndk_http_module.so;
load_module modules/ngx_http_lua_module.so;

events {
    worker_connections  1024;
}

http {

    keepalive_timeout  65;
    upstream test.ase.pi {
       server IP:PORT max_fails=1 max_conns=100 fail_timeout=10;
       server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup;
       keepalive 32;
#      keepalive_timeout 3600s; # NOT allowed < 1.15.3
   }

    upstream introspect_server {
        server IP:PORT max_fails=1 max_conns=100 fail_timeout=10;
        server IP:PORT max_fails=1 max_conns=100 fail_timeout=10 backup;
        keepalive 32;
    }

    lua_shared_dict cache_dict 128m;

server {
    listen              4443 ssl bind;
    server_name         localhost;
    ssl_certificate     /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    ssl_password_file   /etc/nginx/ssl/password_file;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    resolver 8.8.8.8 ipv6=off;
    ase_integration_request;
    ase_integration_response;

    # Set OAuth Client details

    # Set env variable PF_ID &PF_SECRET
    set_by_lua $client_id 'return os.getenv("PF_ID")';
```

Copyright © 2025 Ping Identity Corporation

```
    set_by_lua $client_secret 'return os.getenv("PF_SECRET")';

    # Uncomment next 2 lines to set client credentials here.
    # set $client_id nginx_client;
    # set $client_secret nginx_secret;

    set $oauth_username_key Username;
    set $oauth_client_id_key ClientID;

    # Set the token parameter name below after $arg_ and inside /ase/request.
    set $oauth_key_param $arg_access_token;
    set $oauth_token_param $arg_access_token;

    # Set cache lifetime, default is 120s.
    set $oauth_cache_timeout 120;

    # Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

    location /introspect {
        internal;
        proxy_method    POST;
        if ($arg_auth_token) {
            set $auth_token $arg_auth_token;
        }
        if ($http_authorization ~* .*?(bearer)(\s+)([-a-zA-Z0-9._~+/]+)(,|\s|$)) {
            set $auth_token $3;
        }
        proxy_set_header  Content-Type "application/x-www-form-urlencoded";
        proxy_set_body  "client_id=${client_id}&client_secret=${client_secret}&token=$
{auth_token}";
        proxy_pass_request_body off;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_pass      $oauth_url;
    }

    location /shop {
       include /etc/nginx/pi-pf.conf;
       proxy_pass http://18.209.173.37:4100/shop;

    }
    #DO NOT EDIT BELOW VARIABLE
    set $correlationid $pid-$request_id-$server_addr-$remote_addr-$remote_port-
$request_length-$connection;
    #Certificate location of ASE
    set $certificate /etc/nginx/ssl/test.ase.pi;
    #ASE Token for sideband authentication
    set $ase_token <ASE_TOKEN>;
    #Host header which should be send to ASE
    set $ase_host test.ase.pi;
    #SNI value to use for ASE
    set $ase_ssl_host test.ase.pi;
    #ASE Request Proxy Configuration
    location = /ase/request {
```


Copyright © 2025 Ping Identity Corporation

```
        internal;
        ase_integration https://test.ase.pi;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        # set token key here.
        ase_integration_token_key access_token;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
    #ASE Response Proxy Configuration
    location = /ase/response {
        internal;
        ase_integration https://test.ase.pi;
        ase_integration_method "POST";
        ase_integration_http_version 1.1;
        ase_integration_ase_token $ase_token;
        ase_integration_correlation_id $correlationid;
        ase_integration_host $ase_host;
        ase_integration_ssl_trusted_certificate $certificate;
        ase_integration_ssl_verify    off;
        ase_integration_ssl_verify_depth 1;
        ase_integration_ssl_server_name off;
        ase_integration_ssl_name $ase_ssl_host;
        ase_integration_next_upstream error timeout non_idempotent;
    }
}

}
```

PingAccess sideband integration

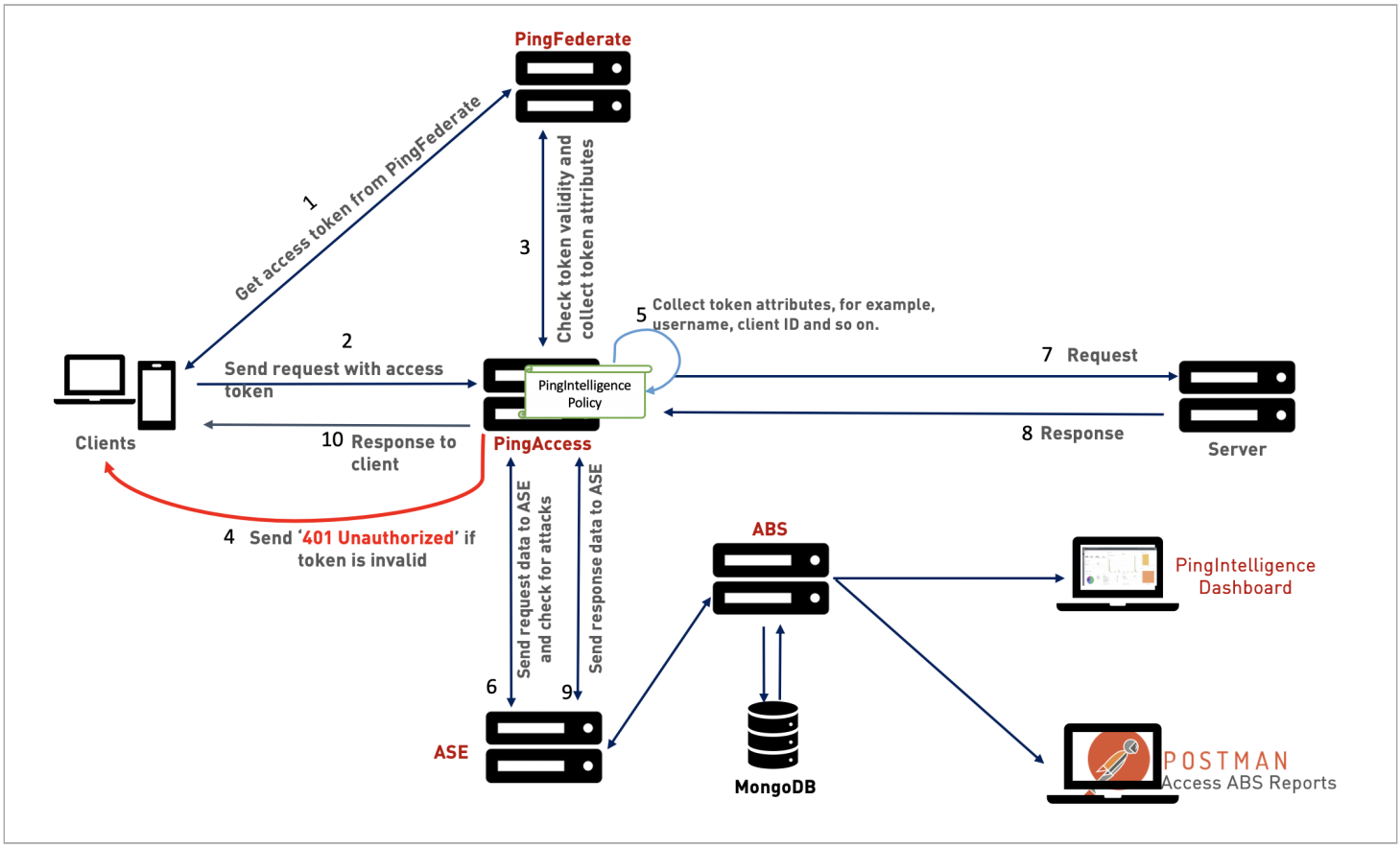
This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with PingAccess.

A PingIntelligencee policy is installed in PingAccess and passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.

The PingIntelligence sideband policy supports interception of OAuth tokens that come as part of a query string. It also supports optional enablement of asynchronous mode to API Security Enforcer (ASE).

The following diagram depicts the architecture of PingIntelligence for APIs components along with PingAccess and PingFederate.

Copyright © 2025 Ping Identity Corporation



The following is the traffic flow through the PingAccess and PingIntelligence for APIs components:

1. The client requests and receives an access token from PingFederate.
2. The client sends a request with the access token received from PingFederate.
3. PingAccess verifies the authenticity of the access token with PingFederate.
4. If the token is invalid, PingAccess returns a 401-unauthorized message to the client.
5. If the token is valid, the PingIntelligence policy running in PingAccess collects API metadata and token attributes.
6. PingAccess makes an API call to send the request information to ASE. ASE checks the request against a registered set of APIs and checks the client identifiers such as IP addresses, cookies against the artificial intelligence (AI)-generated deny list. If all checks pass, ASE returns a `200-OK` response to the PingAccess. If not, a `403-forbidden` response code is sent to PingAccess. The request information is also logged by ASE and sent to the API Behavioral Security (ABS) AI engine for processing.
7. If PingAccess receives a `200-OK` response from ASE, it forwards the request to the backend server. Otherwise, the gateway optionally blocks the client. In synchronous mode, the gateway waits for a response from ASE before forwarding the request to backend server. However, if asynchronous mode is enabled, the gateway forwards the request to the backend server without waiting for the response from ASE. The ASE passively logs the request and forwards it to ABS for attack analysis. It performs attack detection without blocking of attacks.
8. The response from the backend server is received by PingAccess. PingAccess sends the response received from the backend server to the client.

Copyright © 2025 Ping Identity Corporation

9. PingAccess makes a second API call to pass the response information to ASE, which sends the information to the ABS AI engine for processing. ASE receives the response information and sends a `200-OK` to PingAccess.
10. PingAccess sends the response to the client.

Preparing to deploy the PingIntelligence policy

Complete the following steps before configuring PingAccess.

About this task

Before configuring PingAccess:

Steps

1. Confirm the PingAccess version.

   The PingIntelligence policy supports PingAccess versions 5.x and 6.x. If you are using any other version, contact Ping Identity support.

2. Install and configure the PingIntelligence software.

   For more information, see PingIntelligence automated deployment for virtual machines and servers and PingIntelligence manual deployment.

3. Verify that ASE is in sideband mode by running the following command in ASE command line:

   ```
   /opt/pingidentity/ase/bin/cli.sh status
   API Security Enforcer
   status                  : started
    mode : sideband
   http/ws                 : port 80
   https/wss               : port 443
   firewall                : enabled
   abs                     : disabled, ssl: enabled
   abs attack              : disabled
   audit                   : enabled
   sideband authentication : disabled
   ase detected attack     : disabled
   attack list memory      : configured 128.00 MB, used 25.61 MB, free 102.39 MB
   google pubsub           : disabled
   log level               : debug
   timezone                : local (UTC)
   ```

   Troubleshooting:

   If ASE is not in `sideband` mode, then stop ASE and change the mode by editing the `/opt/pingidentity/ase/config/ase.conf` file. Set `mode` as `sideband` and start ASE.

4. For secure communication between PingAccess and ASE, enable sideband authentication by entering the following ASE command:

Copyright © 2025 Ping Identity Corporation

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

5. To generate the token, enter the following ASE command and save the generated authentication token for further use.

   A token is required for PingAccess to authenticate with ASE.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

Configuring PingFederate to extract token attributes

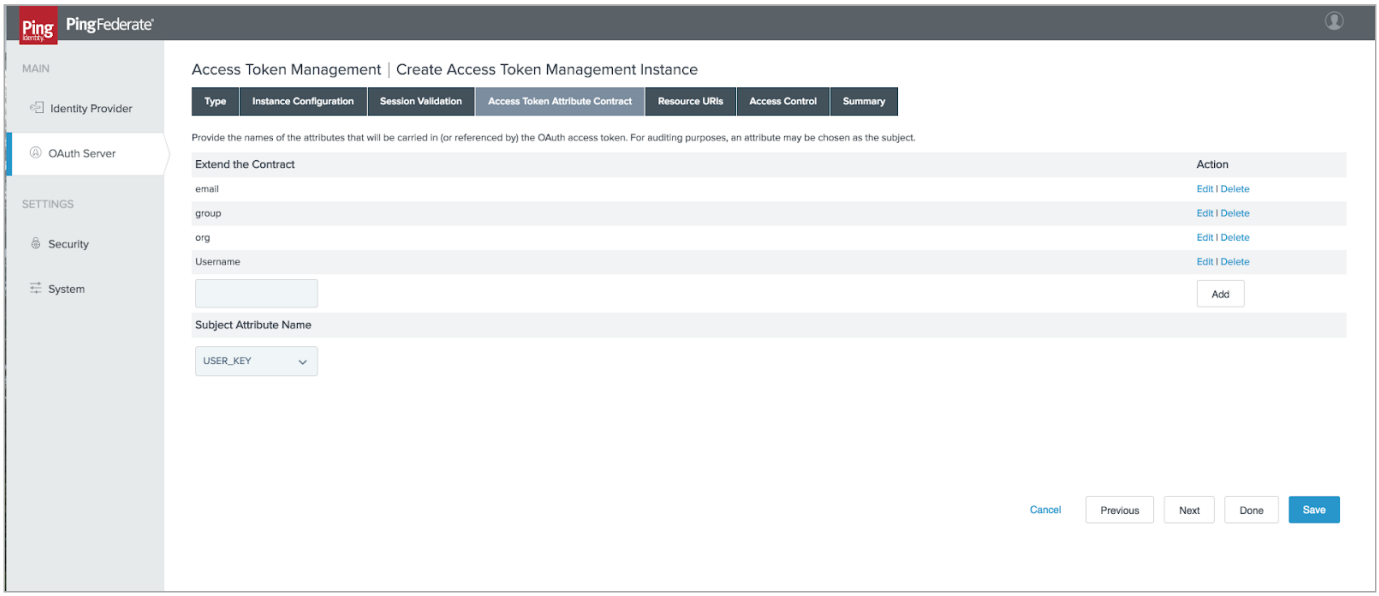
Configure PingFederate for the PingIntelligence policy to be able to extract the username from the incoming token.

About this task

To configure PingFederate to extract token attributes:

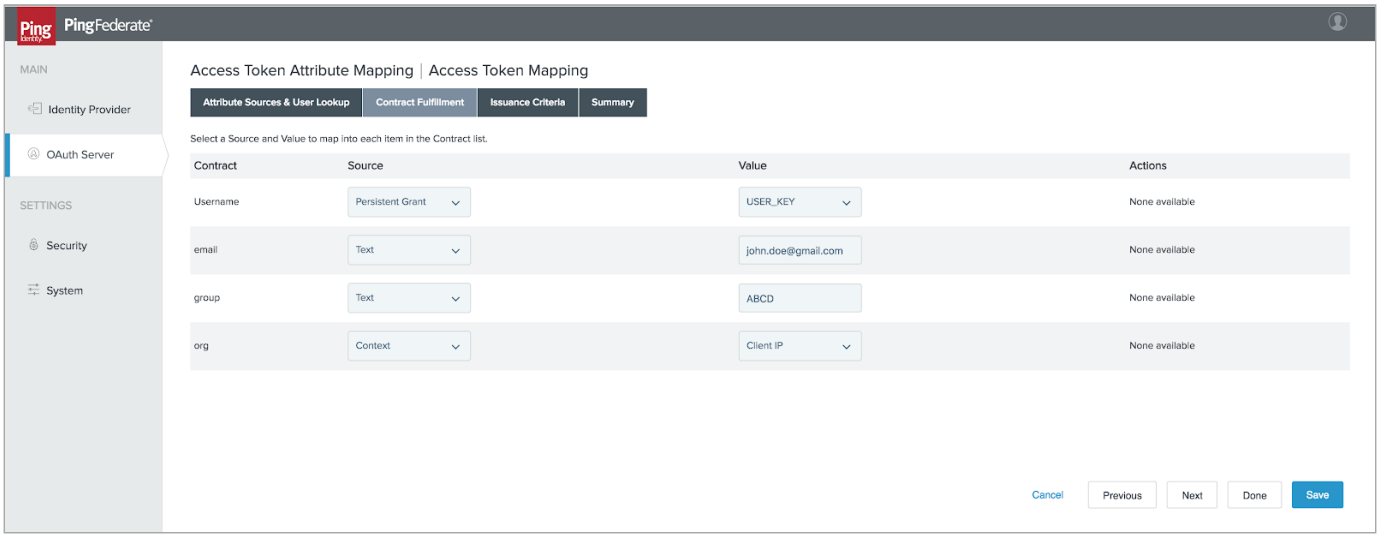
Steps

1. While configuring Access Token Management in PingFederate, add all the attributes that should be exposed for the token. PingFederate provides these attribute values to PingAccess for OAuth tokens.
2. Click Access Token Attribute Contract under Create Access Token Management Instance and add the required attributes. Make sure to at least add Username.



3. After adding the required attributes, configure the attribute sources:
   1. Click Access Token Mapping.
   2. Select the relevant Context.
   3. Click Contract Fulfilment.

Copyright © 2025 Ping Identity Corporation



Deploying the PingIntelligence policy

Integrate PingAccess with the PingIntelligence components.

About this task

> **Note**
>
> We recommend that you increase the default heap size in PingAccess before deploying the PingIntelligence policy for PingAccess 6.x. Refer to the instructions in Modifying the Java heap size for changing the default heap size. For more information, contact Ping Identity support.

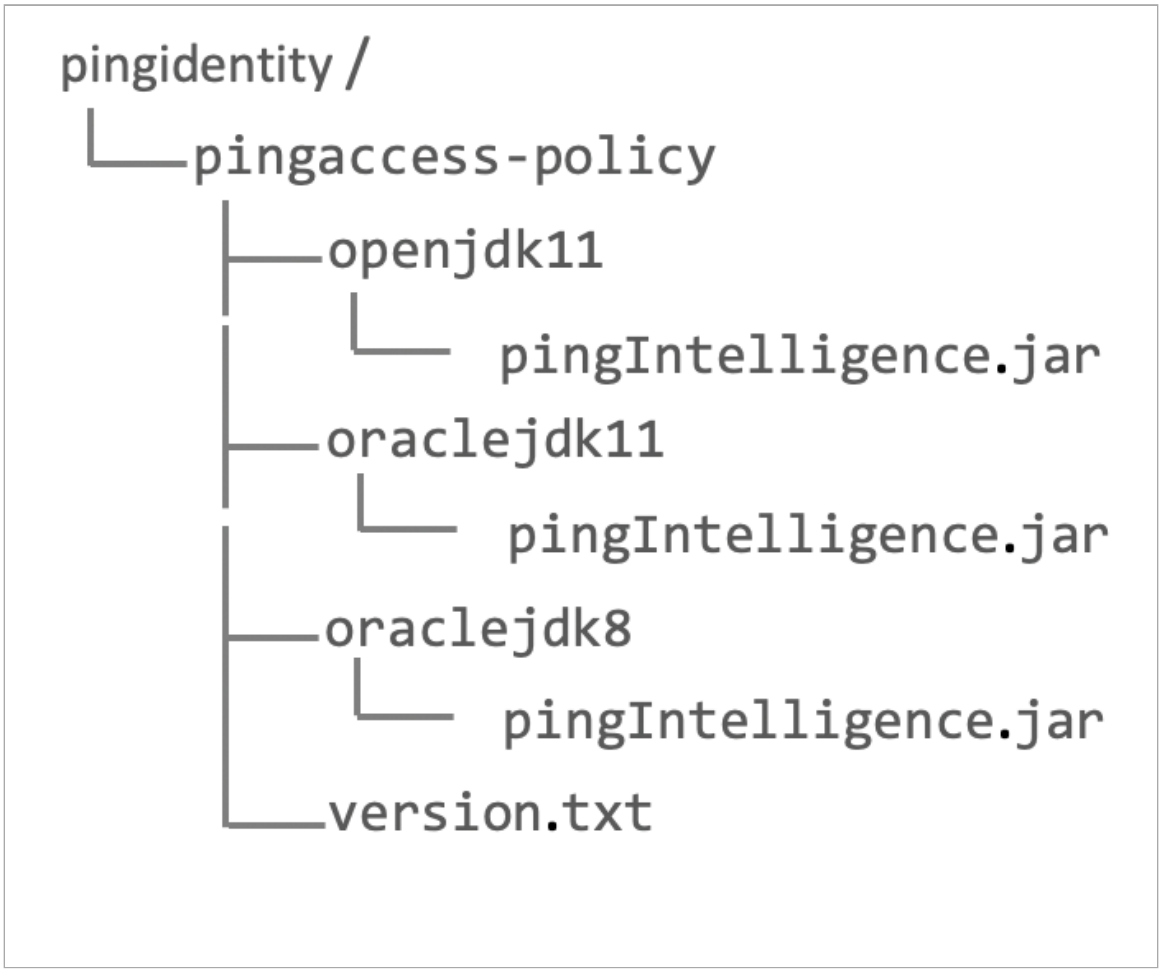
To integrate PingAccess with the PingIntelligence components:

Steps

1. **Download the PingIntelligence policy from the Ping Identity download site and unzip it.**

   The zip file contains three policy files based on the Java Development Kit (JDK) version. Use the policy based on your deployment environment.

Copyright © 2025 Ping Identity Corporation

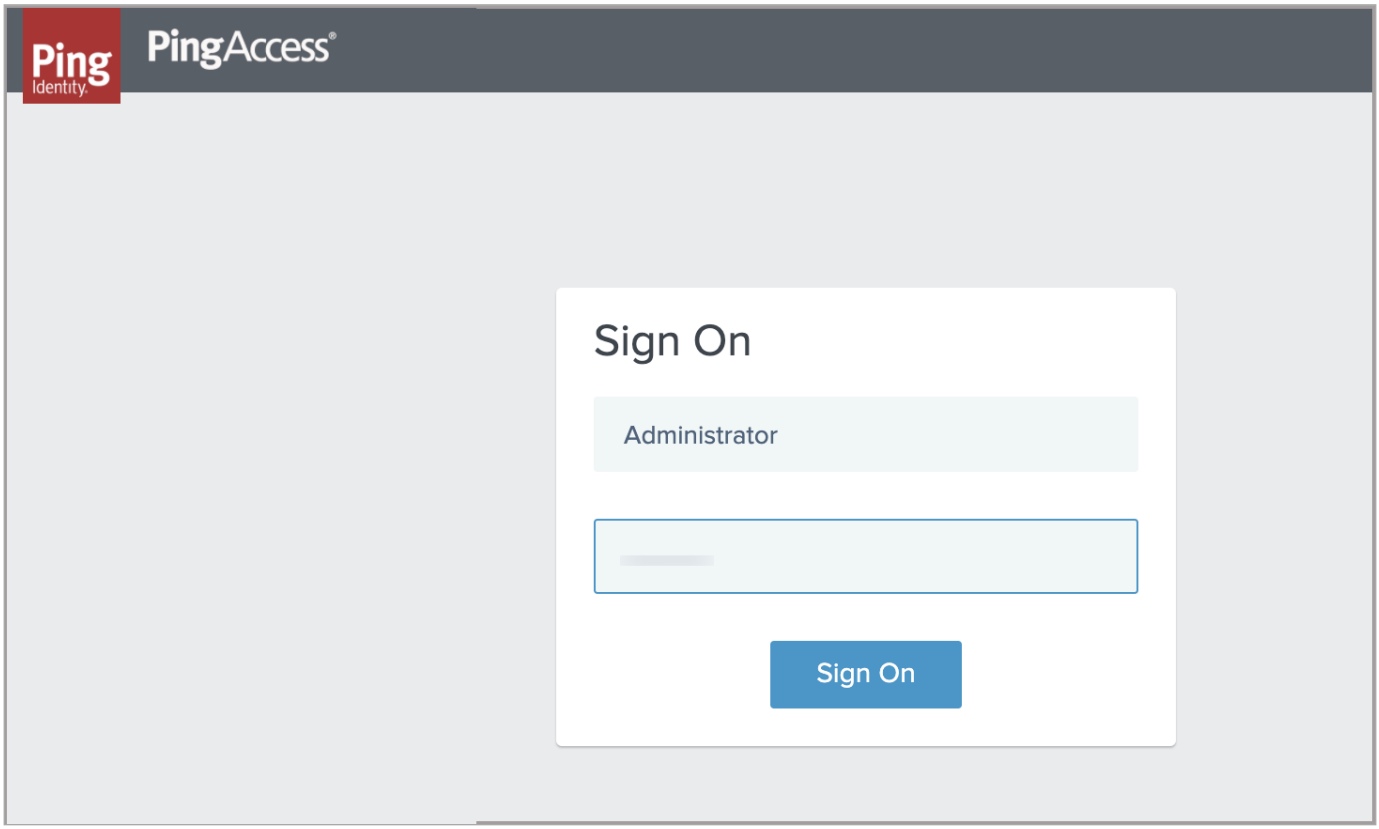


2. Copy the `PingIntelligence.jar` file into the `lib` directory in `PA_home`.

3. Restart PingAccess.

4. Sign on to PingAccess.

Copyright © 2025 Ping Identity Corporation



> **Note**
>
> To support fail-over, a secondary ASE is provisioned. Complete the following steps for both primary and secondary ASEs.

5. **Add the primary ASE as a third-party service:**

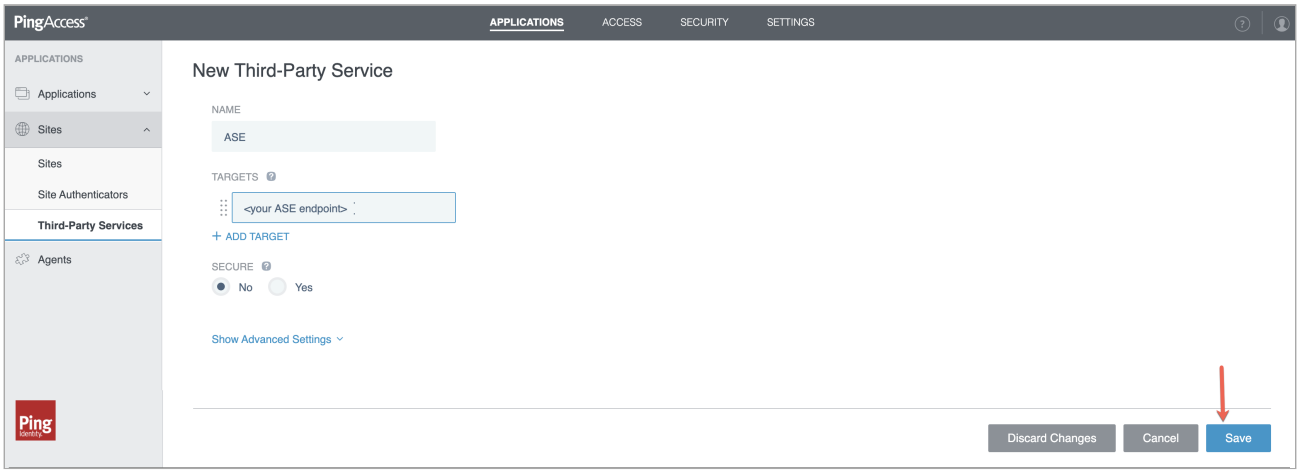
    1. **In the left pane, click Sites.**
    2. **Navigate to THIRD-PARTY SERVICES and click + Add Third-Party Service to add the Primary ASE.**

    3. **In the New Third-Party Service page, enter a name that identifies the Primary ASE in NAME and enter the endpoint used to reach the Primary ASE inTARGET.**

> **Note**
>
> Select options under **SECURE** to connect PingAccess to PingIntelligence ASE using HTTPS.

    4. **Click Save.**

Copyright © 2025 Ping Identity Corporation



6. Repeat step 5 to add the secondary ASE as a third-party service. Enter the name and endpoint specific to the secondary ASE.
7. Add PingIntelligence sideband rule:
    1. In the left pane, click Rules.
    2. In the new Rule page in the NAME field, enter the name of the rule for PingIntelligence.
    3. In the TYPE drop-down list, select PingIntelligence.

    This appears in the drop-down list after adding `PingIntelligence.jar` in `PA_home` in step 2.

    1. Select the ASE endpoint for primary ASE in the PINGINTELLIGENCE ASE ENDPOINT drop-down list.
    2. Select the ASE endpoint for S\secondary ASE inPINGINTELLIGENCE ASE ENDPOINT-BACKUP drop-down list.

    > **Note**
    >
    > If the secondary ASE is not installed, you can choose **Primary ASE Endpoint** in **PINGINTELLIGENCE ASE ENDPOINT-BACKUP** drop-down list.

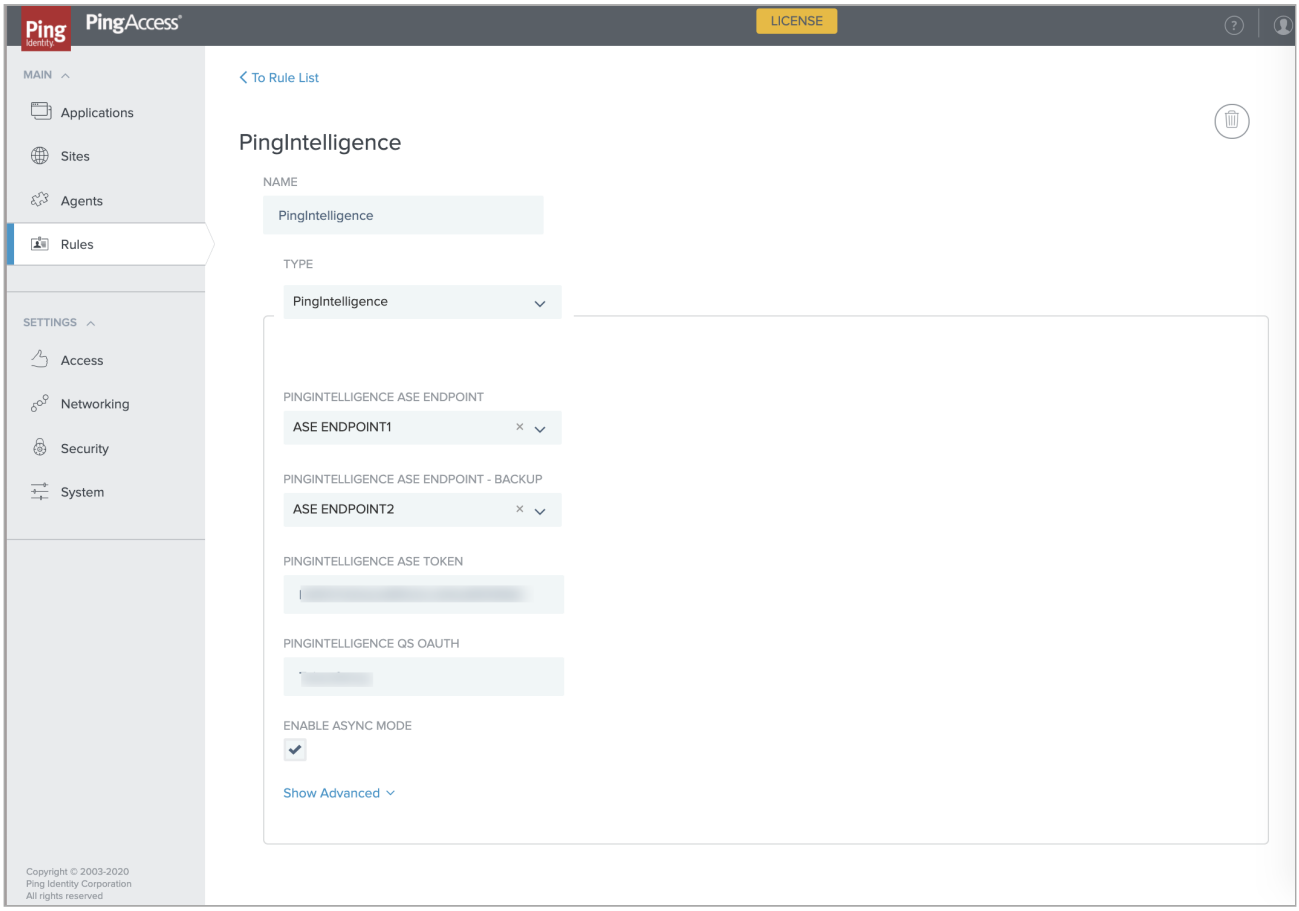
    3. In the PINGINTELLIGENCE ASE TOKEN field, enter the ASE sideband token that was generated for authentication between PingAccess and ASE.
    4. If an OAuth token comes as part of a query string, enter the name of the query string in the PINGINTELLIGENCE QS OAUTH field.

    > **Note**
    >
    > The PingIntelligence policy extracts the OAuth token from the query string configured in **PINGINTELLIGENCE QS OAUTH**. A new Authorization header- `Authorization: Bearer <OAuth token>` is added to the metadata sent to ASE. If there is an existing Authorization header, the token is prepended so that ABS AI engine can analyse it. If the query string has multiple query parameters with the same name, the first parameter is intercepted by the policy.

    5. Select the ENABLE ASYNC MODE to choose Asynchronous mode between PingAccess and ASE.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> The PingIntelligence policy supports both synchronous and asynchronous modes of communication between PingAccess and ASE. By default, the communication mode is synchronous. When the asynchronous mode is enabled, the PingAccess gateway does not wait for a response from ASE and sends the request to backend server. ASE performs attack detection without blocking of attacks in asynchronous mode.

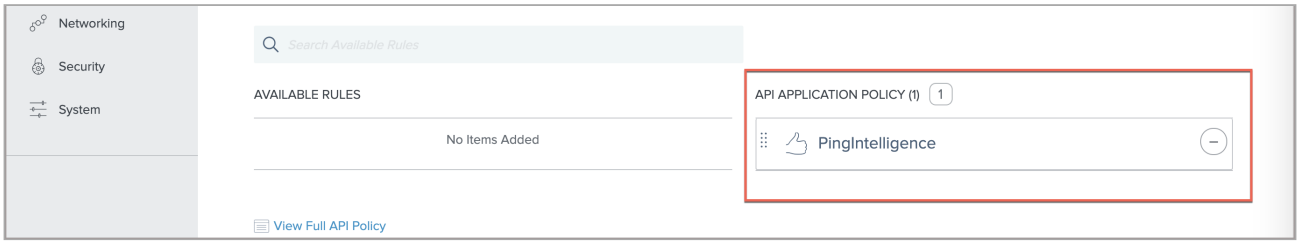


8. **Apply the rule:**

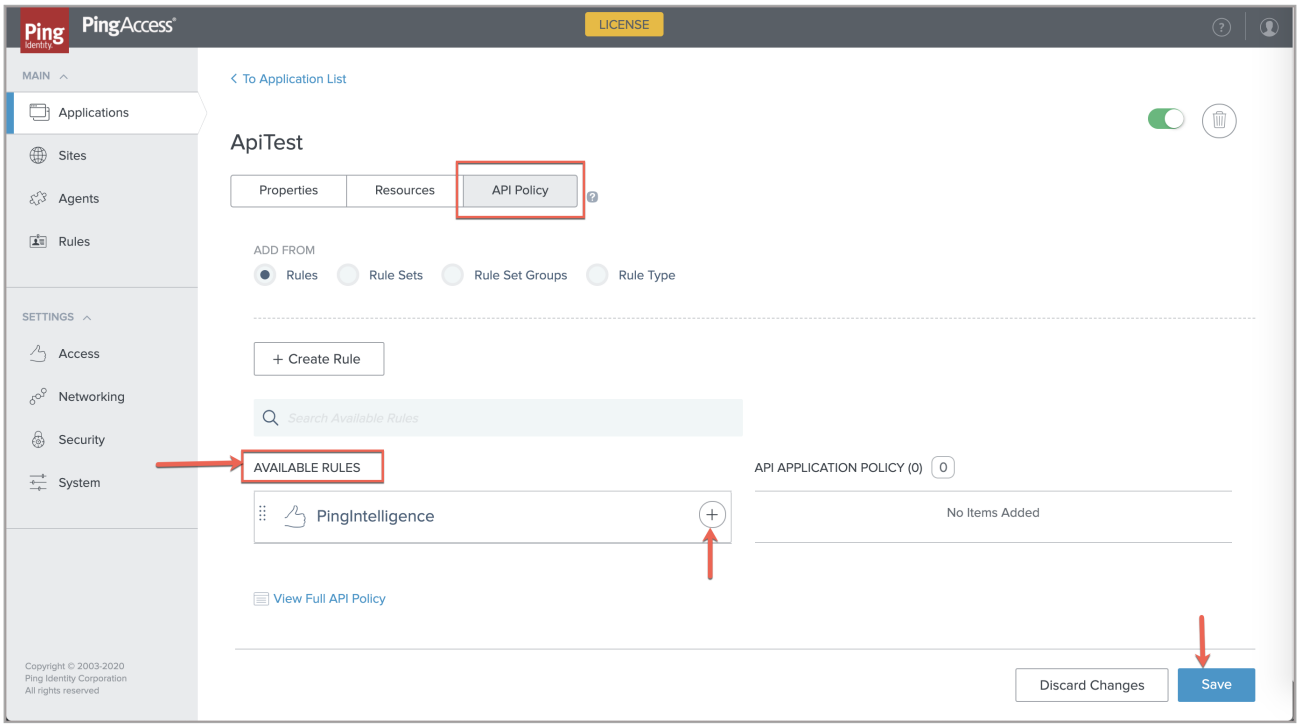
    1. **Edit the existing application.**
    2. **In the edit application page, click API Policy.**
    3. **Under Available Rules, click the ⊕ button for the PingIntelligence rule.**

       Result:

       After clicking the ⊕ button, the PingIntelligence rule moves under the API APPLICATION POLICY as shown in the screen capture below.



4. Click Save to save the rule.



> **Note**
>
> You can selectively apply the PingIntelligence sideband rule to individual resources as well. To apply the sideband rule, click the **RESOURCES** tab and move the rule from **AVAILABLE RULES** onto the policy bar. For more information, see Applying rules to applications and resources.

Optional: Configuring an ASE persistent connection

You can optionally configure ASE TCP keep-alive connections in the `ase.conf` file of ASE.

About this task

To configure an ASE persistent connection:

Steps

- Set `enable_sideband_keepalive` as `true`.

  The default value is set to `false`.

  Example:

Copyright © 2025 Ping Identity Corporation

The following is a snippet of `ase.conf` displaying the default `enable_sideband_keepalive` variable.

```
; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false
```

Configuring API discovery

PingIntelligence API discovery is a process to discover and report APIs from your API environment.

About this task

The discovered APIs are reported in the PingIntelligence Dashboard.

To automatically capture API definitions from PingAccess:

Steps

1. Configure API discovery in the PingIntelligence Dashboard.
    1. Configure the discovery parameters in the Dashboard as outlined in Configure API discovery.

    > **Note**
    >
    > Make sure that the ASE mode is configured to `sideband` in `webgui.properties`, and it matches the configuration in `/pingidentity/ase/config/ase.conf` file in ASE.

    2. Ensure the following configurations specific to PingAccess are set:
        1. Set Discovery source - The Dashboard can discover APIs from three sources, ABS AI engine, PingAccess, and Axway API gateway. The discovery source is configured in the `/pingidentity/webgui/config/webgui.properties file`. Set the `pi.webgui.discovery.source` to `pingaccess`.

        The following is a snippet of the `webgui.properties` file for configuring the discovery source:

        ```
        # api discovery properties
        # discovery source
        # valid values: abs, axway and pingaccess
        # for axway and pingaccess, see config/discovery.properties
         pi.webgui.discovery.source=pingaccess
        ```

        2. Set Credentials - When the API discovery source is PingAccess, configure the gateway management URL and credentials in the `/pingidentity/webgui/config/discovery.properties` file.

        The following is a snippet of the `discovery.properties` file for configuring the credentials:

Copyright © 2025 Ping Identity Corporation

```
# PingAccess config. Only valid if pi.webgui.discovery.source=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Username
# Admin password
pingaccess.management.password=Password
```

2. Configure API discovery in PingAccess:

    1. For the PingIntelligence Dashboard to automatically discover the APIs, include the following parameters in the DESCRIPTION section of an existing application or while you add a new application in PingAccess.

        The application type must be API.

        ```
        {
        "ping_ai": true,
        "ping_host": "",
        "ping_url": "",
        "ping_login": "",
        "ping_cookie": "JSESSIONIDTEST",
        "apikey_qs": "X-API-KEY",
        "apikey_header": "",
        "ping_decoy": false,
        "oauth2_access_token": false,
        "ping_blocking": true
        }
        ```

        The following table describes the parameters captured when the PingIntelligence Dashboard fetches the API definition from PingAccess and adds it to ASE.

Parameter	Description
`ping_ai`	When `true`, PingIntelligence processing is applied to this API. Set to `false` for no PingIntelligence processing. The default value is `true`.
`ping_host`	Hostname of the API. You can configure `*` as `hostname` to support any hostname.
`ping_url`	The base URL of the managed API, for example, `/shopping`. This field cannot be empty.
`ping_login` />	Sign-on URL for the API. The field can be empty.
`ping_cookie` />	Cookie name for the API. The field can be empty.
`apikey_qs`	When API Key is sent in the query string, ASE uses the specified parameter name to capture the API key value. This field can be empty.

Copyright © 2025 Ping Identity Corporation

`apikey_header`	When API Key is part of the header field, ASE uses the specified parameter name to capture the API key value. This field can be empty.
`ping_decoy`	When `true`, API is a decoy API. The values can be `true` or `false`.
`oauth2_access_token`	When `true`, PingIntelligence expects an OAuth token. The values can be true or false.
`ping_blocking`	When `true`, enable PingIntelligence blocking when attack are detected on the API. The default value is `true`. To disable blocking for the API, set to `false`.

Next steps

For more information, refer to:

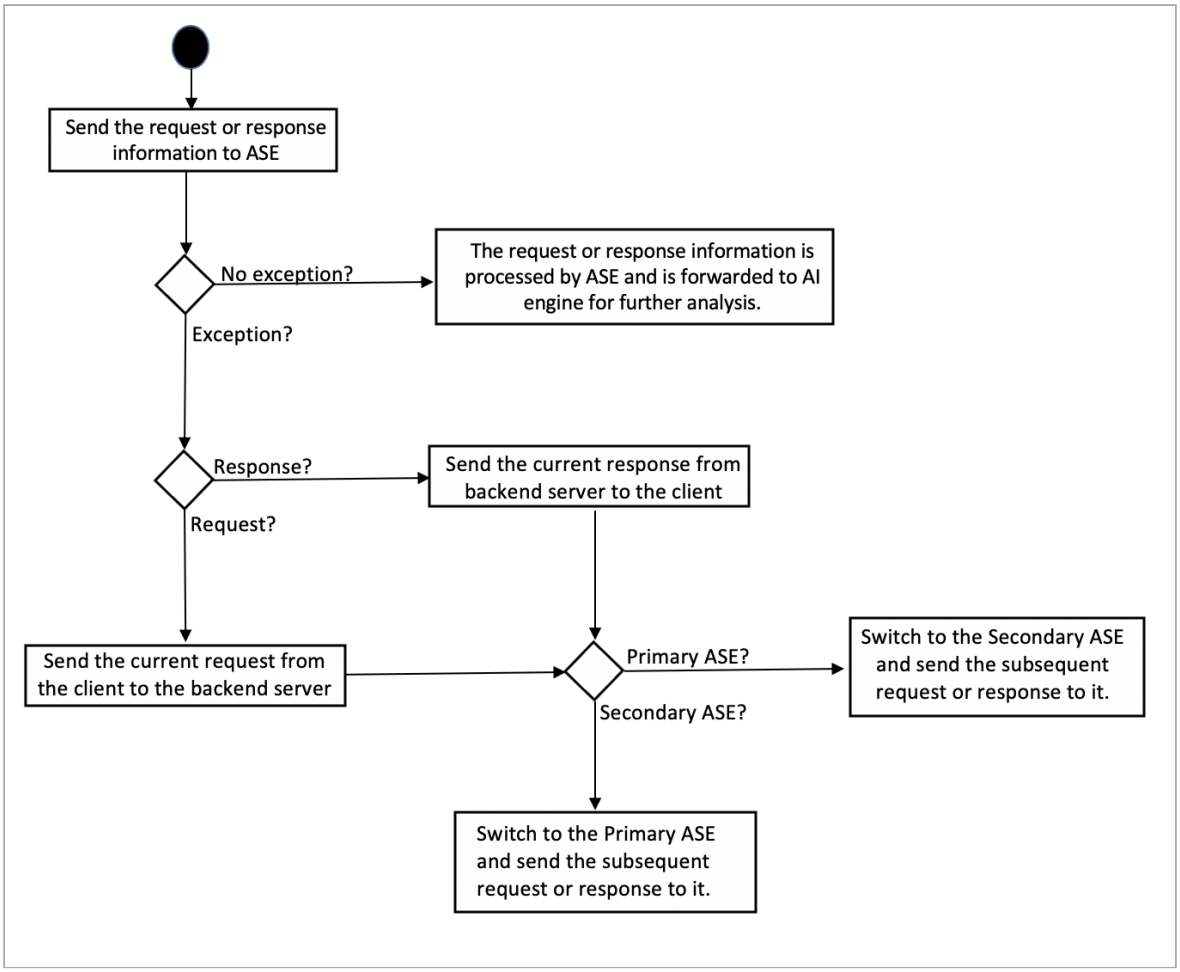
- Discovered APIs
- API discovery and configuration
- Defining an API using API JSON configuration file in sideband mode

Handle exceptions

Learn about exception handling by the PingIntelligence policy when ASE is unavailable.

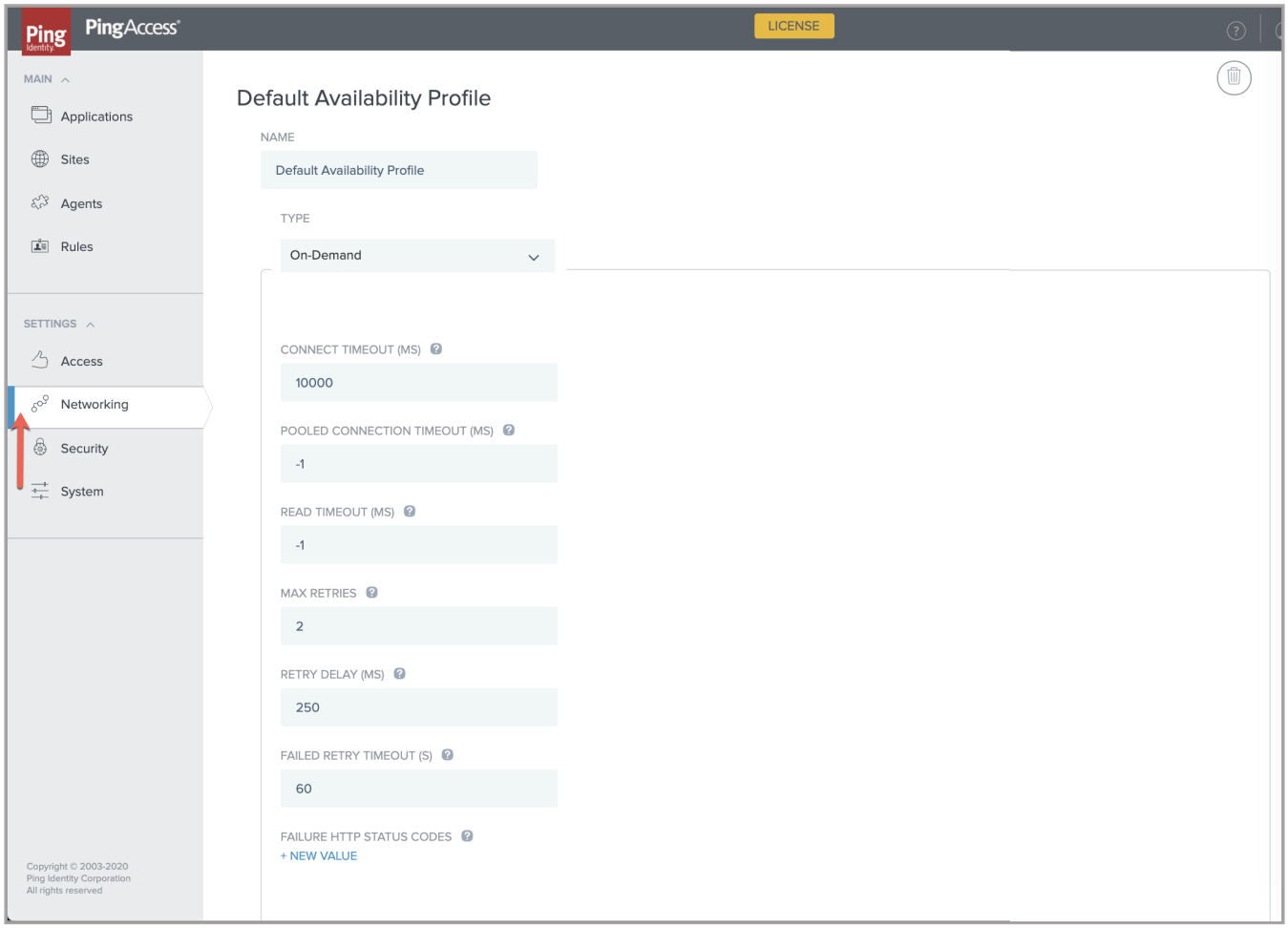
To ensure high availability, the policy supports primary and secondary ASEs. In the event of an exception, the gateway processes the current request or response to the corresponding destination. From the subsequent request or response, a switch happens between the ASEs and the metadata is routed to the other ASE. The following diagram shows the flow when an exception occurs.

Copyright © 2025 Ping Identity Corporation



You can configure an availability profile to define the way PingAccess manages network requests. For more information, see Availability profiles.

Copyright © 2025 Ping Identity Corporation



PingFederate sideband integration

You can deploy PingIntelligence for APIs in a sideband configuration with PingFederate server.

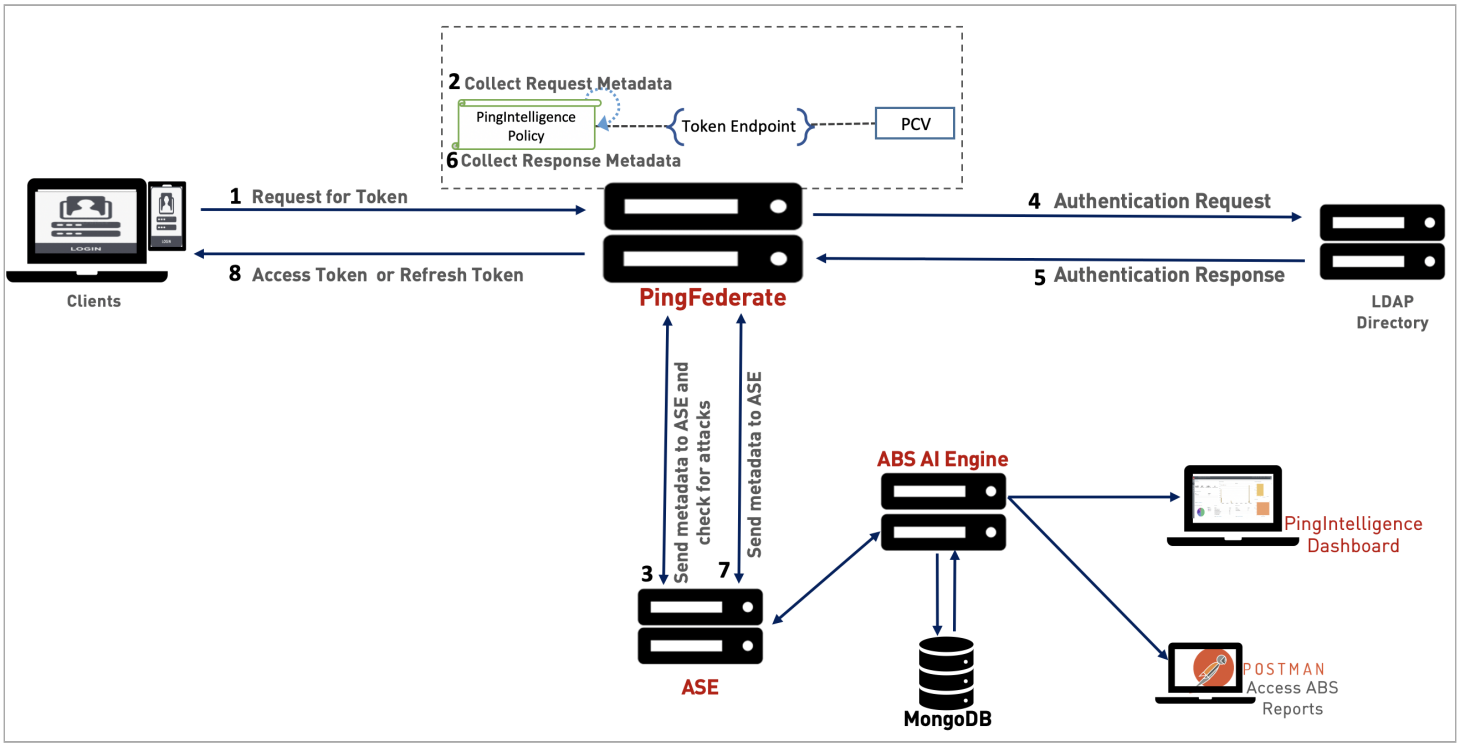
PingIntelligence provides a sideband policy that extracts metadata from an authentication request or response processed by PingFederate. This metadata is passed to PingIntelligence to detect anomalous behavior and attacks by the client. PingIntelligence provides key metrics and forensics around such attacks. It also gives insights into normal traffic patterns by providing detailed client activity reports

The PingIntelligence policy for PingFederate is executed when a client requests an access token or refresh token from PingFederate. The policy secures the token endpoint `/as/token.oauth2`. For more information on the OAuth endpoints exposed by PingFederate, see PingFederate OAuth 2.0 endpoints.

The PingIntelligence policy supports attack detection and reporting based on IP addresses of the clients. It is deployed in PingFederate as a servlet filter. It supports both OpenID Connect (OIDC) and Security Assertion Markup Language (SAML) V2 standards. The policy deployment does not require any reconfiguration of password credential validator (PCV).

The following diagram shows the architecture of PingIntelligence for APIs components and the interaction flow with PingFederate. The Lightweight Directory Access Protocol (LDAP) directory component in the diagram is used for illustrative purposes. PingFederate also supports other directories and user data-stores through PCVs. For more information, see Password Credential Validators.

Copyright © 2025 Ping Identity Corporation



The traffic flow through the PingFederate and PingIntelligence for APIs components is as follows:

1. A client sends a request with its authorization grant to PingFederate to obtain an access or refresh token.
2. The PingIntelligence policy deployed in PingFederate intercepts this request and extracts metadata such as origin IP address, and so on.
3. PingFederate makes an API call to send the metadata to API Security Enforcer (ASE). ASE checks the client identifiers such as IP addresses against its blacklist. A blacklist is a list of client identifiers that were detected executing an attack. If all checks pass, ASE returns a `200-OK` response to PingFederate. If the checks do not pass, ASE sends a `403-Forbidden` response code to PingFederate and optionally blocks the client. In both the cases, ASE logs the request information and sends it to the API Behavioral Security (ABS) AI engine to analyze the traffic patterns.
4. PingFederate forwards the client authentication request to the supported directory server.
5. PingFederate receives the response from the server.
6. The PingIntelligence for APIs policy intercepts the response and extracts the metadata.
7. PingFederate makes a second API call to pass the response information to ASE, which sends the information to the ABS AI engine for processing.
8. PingFederate sends the requested token to the client.

Related links

- Sideband ASE
- ABS AI Engine
- PingIntelligence for APIs Dashboard

Copyright © 2025 Ping Identity Corporation

Preparing for deployment

Complete the following prerequisites before deploying PingIntelligence policy on PingFederate

Before you begin

- Verify versions supported. The PingIntelligence policy is qualified with the following combination.

PingFederate PingFederateVersion	JDK version	Password Credential Validator
PingFederate 9.3.3	Oracle JDK8.0.u261	◦ OpenLDAP-2.4.44 ◦ Simple username password credential validator (PCV)

> **Note**
>
> If you are using any other versions of PingFederate or JDK, or any other PingFederate-supported PCV, contact the Ping Identity support team for deployment support.

- Install and configure PingIntelligence software. For more information on PingIntelligence deployment, see PingIntelligence automated deployment or PingIntelligence manual deployment.

About this task

To prepare for deployment of the PingIntelligence policy:

Steps

1. Verify that API Security Enforcer (ASE) is in sideband mode by running the following ASE command:

```
/opt/pingidentity/ase/bin/cli.sh status
```

Result:

If ASE is in sideband mode, you will see the following result:

```
API Security Enforcer
status                  : started
 mode : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : disabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.61 MB, free 102.39 MB
google pubsub           : disabled
log level               : debug
timezone                : local (UTC)
```

Copyright © 2025 Ping Identity Corporation

Troubleshooting:

If ASE is not in sideband mode, complete the following steps:

1. Stop ASE if it is running. For more information, see Starting and stopping ASE.
2. Navigate to `/opt/pingidentity/ase/config/`.
3. Edit the `ase.conf` file and set `mode` parameter to `sideband`.
4. Start ASE. For more information, see Starting and stopping ASE.

2. For a secure communication between PingFederate and ASE, enable sideband authentication by entering the following ASE command:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

3. Generate a sideband authentication token by entering the following ASE command.

   A token is required for PingFederate to authenticate with ASE.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

4. Save the generated authentication token for further use.
5. Enable connection keepalive between PingFederate and ASE.
   1. Stop ASE if it is running. For more information, see Starting and stopping ASE.
   2. Navigate to `/opt/pingidentity/ase/config/`.
   3. Edit the `ase.conf` file and set `enable_sideband_keepalive` parameter to `true`.
   4. Start ASE. For more information, see Starting and stopping ASE.

Deploying the PingIntelligence policy

Deploy the PingIntelligence for APIs policy in PingFederate and complete the required configurations.

Before you begin

Complete the configuration steps in Preparing for deployment.

About this task

To deploy the PingIntelligence policy:

Steps

1. Download the PingIntelligence for APIs policy file from the Sideband Integrations section of the PingIntelligence for APIs download page and copy it to the node hosting PingFederate server.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> If the PingFederate server is deployed in a cluster, then copy the policy to all the runtime engine nodes of the cluster.

2. Extract the policy file by entering the following command.

```
$ untar pi-api-pf-policy-4.3.tar.gz
```

3. Stop PingFederate. For more information, see Start and stop PingFederate.
4. Copy the policy to the `pingfederate/server/default/deploy` directory.

```
$ cp pingidentity/pf-policy/pf-pi4api-filter.jar  <pf_install>/pingfederate/server/default/deploy/
```

5. Complete the following configurations:
   - Configuring the PingIntelligence servlet filter
   - Configure API JSON in ASE
6. Start PingFederate. For more information, see Start and stop PingFederate.

Configuring the PingIntelligence servlet filter

Configure the servlet filter for the PingIntelligence policy in the `webdefault.xml` file in PingFederate.

About this task

To define the PingIntelligence for APIs servlet filter:

Steps

1. Add the the following filter configuration to the `<pf_install>/pingfederate/etc/webdefault.xml` file. Add the filter configuration within the `<web-app></web-app>` element.

> **Note**
>
> If there are multiple filters in the `webdefault.xml` file, then place `pi4APIFilter` at the end.

Copyright © 2025 Ping Identity Corporation

```
<filter>
    <filter-name>pi4APIFilter</filter-name>
    <filter-class>com.pingidentity.pi.servlets.PI4APIServletFilter</filter-class>
    <init-param>
        <param-name>ASE-Primary-URL</param-name>
        <param-value>https://<IP address of primary ASE>:<Port number></param-value>
    </init-param>
    <init-param>
        <param-name>ASE-Secondary-URL</param-name>
        <param-value>https://<IP address of secondary ASE>:<Port number></param-value>
    </init-param>
    <init-param>
        <param-name>ASE-Token</param-name>
        <param-value><ASE authentication token></param-value>
    </init-param>
    <init-param>
          <param-name>Enable-Blocking</param-name>
          <param-value>false</param-value>
      </init-param>
  </filter>
  <filter-mapping>
    <filter-name>pi4APIFilter</filter-name>
    <url-pattern>/as/token.oauth2</url-pattern>
  </filter-mapping>
```

2. Make sure the following configurations are set correctly:
   - The `filter-class` element is configured to `com.pingidentity.pi.servlets.PI4APIServletFilter`.
   - The `pi4APIFilter` is mapped to the token endpoint URL of PingFederate by configuring the `url-pattern` element to `/as/token.oauth2`.
   - The `filter-name` element in both the `<filter>` and `<filter-mapping>` blocks is `pi4APIFilter`.
3. Substitute the actual values for the `init` parameters in the `pi44APIFilter` filter.

   The following table explains the PI4API `init` parameters in detail. The parameteers control the communication with API Security Enforcer (ASE). You can contact Ping Identity support team for the actual values of these parameters.

Parameter Name	Description
`ASE-Primary-URL`	The URL or IP address of the ASE primary host. **Note** To support high availability, PingIntelligence provides ASE primary and secondary nodes.
`ASE-Secondary-URL`	The URL or IP address of the ASE secondary host.

Copyright © 2025 Ping Identity Corporation

Parameter Name	Description
`ASE-Token`	The ASE sideband authentication token. You can obfuscate the sideband authentication token using one of the following utilities available in the PingFederate `<pf_install>/pingfederate/bin/` directory: ◦ On Windows: `obfuscate.bat` ◦ On Linux: `./obfuscate.sh` If you need further assistance in using the utility, contact Ping Identity support.
`Enable-Blocking`	You can optionally block a client that has been detected executing an attack. To block the client, you need to enable blocking in ASE by setting the `Enable-Blocking` to `true`. The default value is `false`.

Configuring the API JSON file

Configure the API JSON file in API Security Enforcer (ASE).

About this task

The API JSON file parameters define the connectivity to the token endpoint.

To configure the API JSON file:

Steps

1. Navigate to the `/pingidentity/ase/config/` directory.
2. Edit the `sideband_api.json.example` file, and set the value of the `url` parameter to `/as` and the `login_url` parameter to `/as/token.oauth2`.

   > **Note**
   >
   > `/as/token.oauth2` is the token endpoint of PingFederate authorization server.

3. Rename the `sideband_api.json.example` file to `pf.json`.
4. After configuring the API JSON file, add it to ASE by executing the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh –u admin -p admin add_api pf.json
   ```

   Example:

   The following is a sample configuration of the API JSON file:

Copyright © 2025 Ping Identity Corporation

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/as",
    "hostname": "*",
    "cookie": "",
    "oauth2_access_token": false,
    "apikey_qs": "",
    "apikey_header": "",
    "login_url": "/as/token.oauth2",
    "enable_blocking": true,
    "api_memory_size": "128mb",
    "decoy_config": {
      "decoy_enabled": false,
      "response_code": 200,
      "response_def": "",
      "response_message": "",
      "decoy_subpaths": []
    }
  }
}
```

Note

For more information on configuring API JSON parameters, see Defining an API using API JSON configuration file in sideband mode.

TIBCO integration

Integrate your TIBCO Cloud Mashery gateway with PingIntelligence.

About this task

The PingIntelligence software components handle all artificial intelligence (AI) processing and dashboard reporting. The component responsible for collecting API traffic metadata is called the API Security Enforcer (ASE). The ASE connects to the PingIntelligence software to deliver the API traffic metadata collected from the TIBCO Cloud Mashery gateways for the AI processing.

PingIntelligence consists of the following components:

- **The ASE is deployed on-premise as a virtual machine or Docker container. It captures the metadata of the monitored APIs and sends it to the PingIntelligence service for processing.**

Note

Payload data is never sent to the cloud. The ASE is deployed in sideband mode with TIBCO Mashery API gateways. It connects to one or more gateways through a sideband policy deployed on the gateways. For more information, seehttps://docs.mashery.com/connectorsguide/GUID-118777DD-8878-4753-871F-8E0E380FAA39.html[Sideband Connector for TIBCO Cloud Mashery].

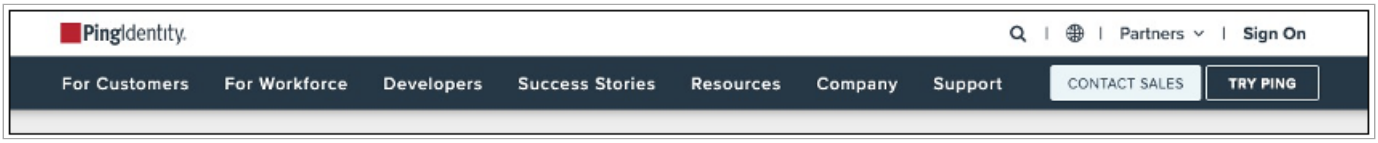
Copyright © 2025 Ping Identity Corporation

- The AI engine processes the metadata sent by ASE to identify new APIs, deliver increased visibility, and detect abnormal API traffic patterns. It builds machine learning models that self-train based on the API traffic. When an abnormal situation or attack needs to be blocked, it communicates with the ASE to block the clients from which the traffic originates.
- The PingIntelligence dashboard provides rich analytics on API activities. It tracks API activity across all API gateway clusters and clouds to deliver a single pane of glass used to monitor the API infrastructure. Newly discovered APIs are surfaced and are tracked once selected. All tokens or cookies used and IP addresses are associated with each user identity. The dashboard provides information on the training status of the APIs and delivers deep insights on abnormal situations and attacks detected.

To integrate your TIBCO Cloud Mashery gateway with PingIntelligence:

Steps

1. Go to https://pingidentity.com and click Try Ping on the right side of the page.



2. To deploy the Mashery connector, see the TIBCO connector documentation.

   For more information on sideband connectivity, see API environment integration with on-premise ASE.

WSO2 integration

PingIntelligence for APIs in a sideband deployment integrates with WSO2 API gateway to provide in-depth analytics on API traffic.

In the deployment, the WSO2 API Gateway is the primary component that intercepts API requests and applies various types of policies. Each policy is executed using an API handler. The API Gateway architecture allows users to add specific handlers to perform various tasks in different stages of the request flow.

This implementation comes with a handler that allows users to perform sideband calls to the PingIntelligence API Security Enforcer (ASE). With these sideband calls, it publishes API request metadata to Ping and checks the validity of the request. It does the same for the response. With the provided request metadata, PingIntelligence ASE can detect abnormal access patterns and build a knowledge base using API request data sent to it.

For more information on PingIntelligence WSO2 integration, see Artificial Intelligence Based API Security with WSO2 and PingIntelligence for APIs.

Copyright © 2025 Ping Identity Corporation

Getting Started with PingIntelligence

API environment integration with on-premise ASE

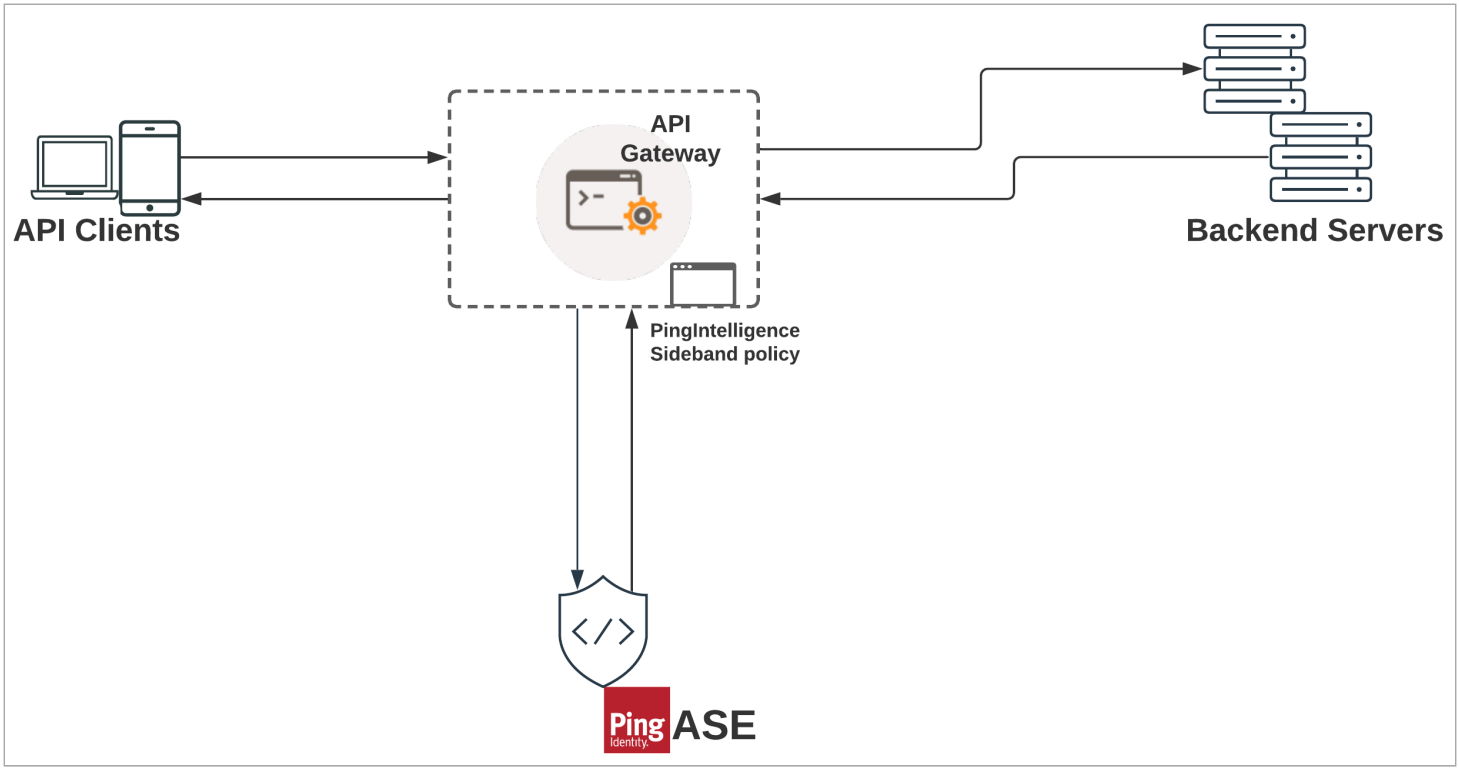
Configure the deployment mode in API Security Enforcer (ASE) and integrate your API environment with ASE.

Ping Identity supports two integration options for your API environment:

- Sideband deployment
- Inline deployment

Sideband deployment

When deployed in sideband mode, ASE receives API calls from an API gateway, which passes API traffic information for artificial intelligence (AI) processing. ASE requires no changes to clients or backend API servers. In sideband deployment, ASE works along with the API gateway to protect your API environment. A custom sideband policy is provided, which is deployed in the gateway to route the API traffic. The following diagram shows ASE in sideband deployment mode.



To configure ASE for a sideband environment, see #/section_hrs_jnh_vqb.

Copyright © 2025 Ping Identity Corporation

Configure ASE for sideband deployment

PingIntelligence provides custom sideband policies for API gateways, routers, and other API platforms to support integration with your API environments. See API gateway integrations supported by PingIntelligence for a list of gateway integrations supported along with the deployment instructions. The sideband policy is deployed in your API gateway, and it sends the request and response API metadata to ASE for processing. Follow the instructions in the integration guides to deploy a sideband policy in your environment.

After you determine which API gateways to integrate, set the deployment mode in the `ase.conf` file located in the `/<ASE installation path>/pingidentity/ase/config/` directory.

Parameter	Description
`mode`	Set the mode to `sideband` for ASE to work in a sideband mode.
`enable_sideband_keepalive`	When set to `true`, ASE sends a keep-alive in response header for the TCP connection between API gateway and ASE. With the default `false` value, ASE sends a connection close in the response header for connection between API gateway and ASE. **Note** Set to `true` unless using a MuleSoft API gateway.
`enable_sideband_authentication`	Set to `true` if you intend to enable authentication between an API gateway and ASE. After setting it to `true`, generate a sideband authentication token using the ASE `create_sideband_token` command. **Note** Set to `false` for evaluation deployments to simplify setup.

Important

After updating the settings, restart ASE using the following commands:

- Change the working directory to `/bin` and run the **`stop.sh`** script:

```
# /<ASE installation path>/pingidentity/ase/bin/stop.sh
```

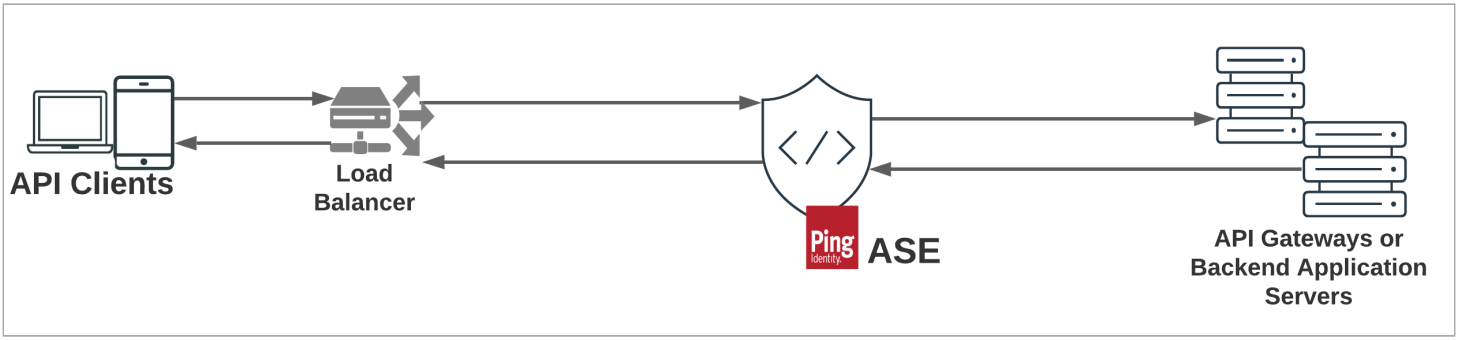
- Change the working directory to `/bin` and run the **`start.sh`** script:

```
# /<ASE installattion path>/pingidentity/ase/bin/start.sh
```

Copyright © 2025 Ping Identity Corporation

Inline environment

When deployed in inline mode, ASE is a reverse proxy deployed between the API clients and servers. It is typically deployed behind load balancers, such as AWS Elastic Load Balancing (ELB), to distribute traffic across an ASE cluster for high availability. ASE terminates SSL connections from API clients and then routes the requests to the destination APIs on an API gateway or app servers, such as Node.js, WebLogic, or Tomcat. The following diagram shows an inline deployment.



> **Note**
>
> To continue with an inline deployment, see Inline ASE.

API discovery in your environment

PingIntelligence supports API discovery in sideband and inline mode as an automated method of building API definitions that provide the properties of the managed APIs.

These API definitions are then used to provide API visibility and detection of anomalous client behavior. You can configure API discovery through the Dashboard, which displays, manages, and renders the discovered APIs. The Dashboard also allows you to edit the discovered APIs and export their definition files.

To set up PingIntelligence for discovery, you must:

1. Configure ASE with API discovery
2. Configure discovery settings in the Dashboard

Configuring ASE with API discovery

About this task

> **Note**
>
> Configure API discovery when API Security Enforcer (ASE) is deployed in sideband mode. To configure ASE for inline discovery, contact Ping Identity support.

ASE requires a `root` definition that enables it to route all API traffic to the AI engine. The AI engine receives and monitors all API traffic that is not associated with a known API. It analyzes the traffic and builds API models for the unknown APIs, which are shown on the Discovery dashboard.

To add a `root` API in ASE:

Copyright © 2025 Ping Identity Corporation

Steps

1. Use the sample `root` API JSON shipped with ASE in the `<ASE_Installation path>/pingidentity/ase/config/api/` directory and configure the API JSON for the `root` API.

   For sideband environments, use the following settings:

Parameter	Setting
`protocol`	`http`
`url`	`/`
`hostname`	`*`

2. To capture client identifiers such as token, cookies, API keys, IP addresses, and username, configure the `root` API JSON file with the following client identifiers.

   > **Note**
   >
   > If the identifiers are not present in at least 50% of the traffic received for a discovered API, then the identifiers are not reported or used in Indicator of Attack (IoA) detection.

Client Identifier	Description
`oauth2_access_token`	If a bearer token is present, set to `true`.
`cookie`	If cookies are used as the primary client identifier, configure the cookie name.
`apikey_qs` or `apikey_header`	Set for the API key in query parameter or for the API key in header.

Example:

The following is a sample API JSON for the `root` API:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",
 "url": "/",
 "hostname": "*",
 "cookie": "",
 "oauth2_access_token": true,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_memory_size": "1mb",
 "decoy_config":
{ "decoy_enabled": false, "response_code": 200, "response_def": "", "response_message": "", "decoy_subpaths":
[] }
}
 }
```

> **Note**
>
> IP addresses and usernames are captured separately.

3. After configuring an API JSON file for the `root` API, add it to ASE to initiate the API discovery process by running the following command:

```
/<ASE_Installation path>/pingidentity/ase/bin/cli.sh –u admin -p admin add_api {file_path/api_name}
```

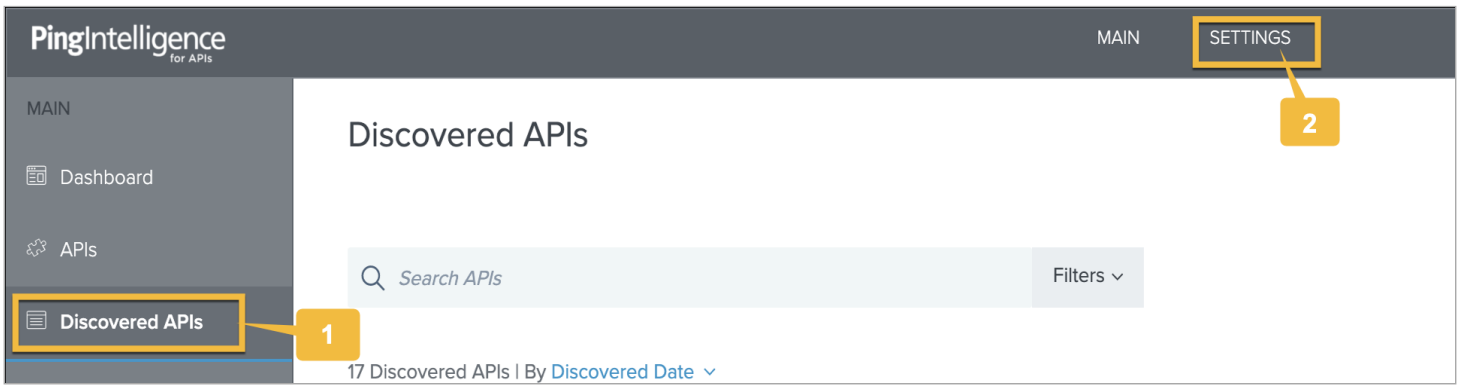
Configuring discovery settings in the Dashboard

About this task

To customize the discovery process, configure the discovery parameters on the Dashboard.

Steps

1. Go to Discovered APIs → Settings.



1. Click the Discovery Configuration tab and set the value for AI Engine Subpath Depth.

Copyright © 2025 Ping Identity Corporation

AI Engine Subpath Depth defines the number of subpaths used to uniquely discover the base path of a new API. The maximum allowed value is 6 when ASE is deployed in inline mode and 10 when ASE is deployed in sideband mode.

Example:

The following are examples of subpath values and what they mean:

- `1` indicates `/atmapp` is the base path for `/atmapp/zipcode`, `/atmapp/update`, and so on.
- `3` indicates `v1/cust1/atmapp` is the base path for `v1/cust1/atmapp/zipcode`, and so on.

2. Click Discovered APIs on the Dashboard and click Export to download the API definition in `.json` format .

3. Add the downloaded API JSON definitions to ASE by running the following command:

```
# /<ASE_Installation path>/pingidentity/ase/bin/cli.sh –u admin -p admin add_api {file_path/
api_name}
```

Copyright © 2025 Ping Identity Corporation

Managing PingIntelligence for APIs

Dashboard

The Dashboard provides a near real-time snapshot of your application programming interface (API) environment.

The Dashboard has insights on user activity, attack information, blocked connections, forensic data, and much more.

You can limit the data to a specific time range, such as the current day, the past week to date, the past month to date, or the past 6 months.

The graph at the top of the page shows global API activity at a glance, including the number of requests and indicators of attack (IoAs) for the specified period.

Click any of the dashboard charts for a more detailed display with filters and options for further inspection and analysis.

Chart Name	Description
Top APIs	Top APIs requests and IoA breakdown.
Blocklist	Distribution of blocklists by client type.
Client IoAs	IoAs management breakdown by client type and client.

Dashboard chart types

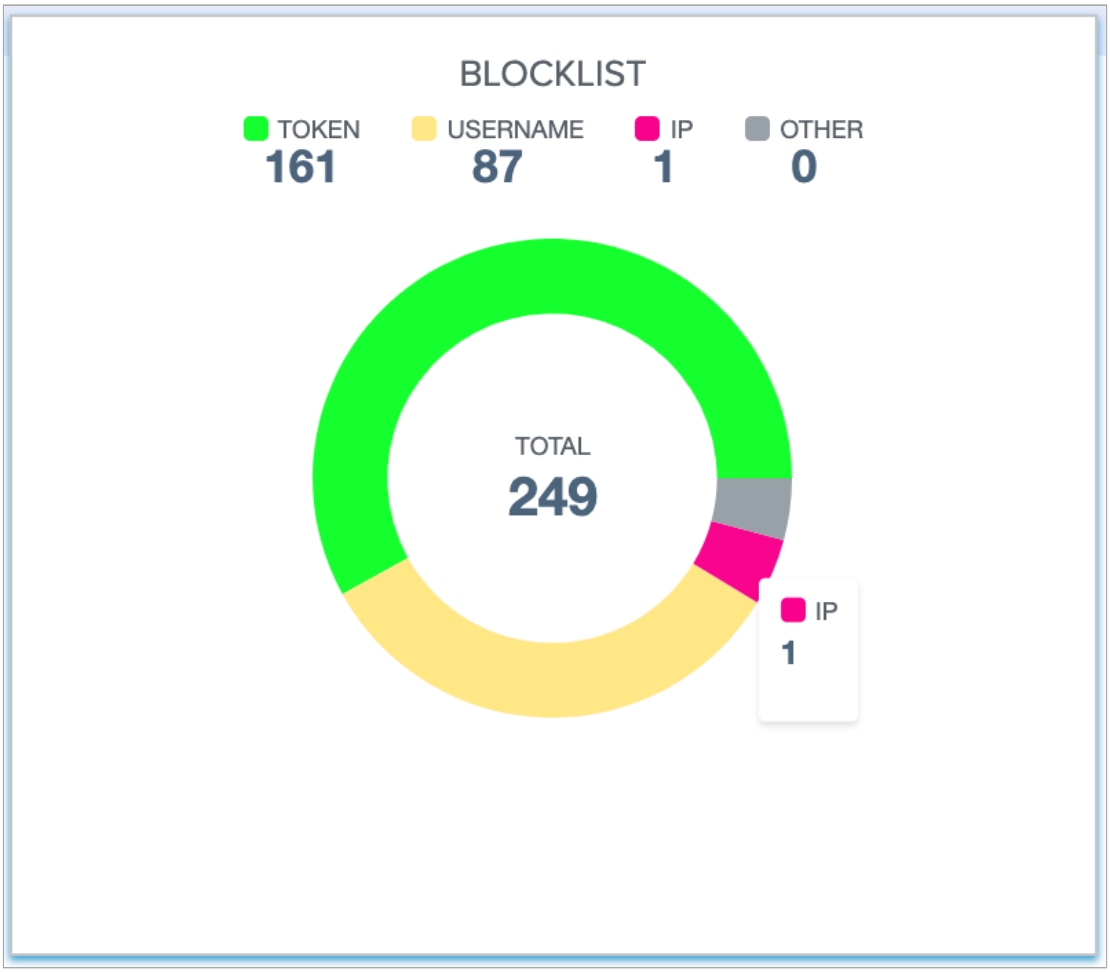
The Dashboard charts are all interactive. You can:

- Hover over any colored area of a pie or bar chart to see additional details.
- Move the mouse horizontally on a line chart to see the plotted value.
- Click the date picker to change the reporting period:
    - Last 6 months
    - Last month
    - Last week
    - Last day
- Use a filter to change a chart appearance. Filtered charts offer additional information, as shown in the pie charts and bar charts examples.

Copyright © 2025 Ping Identity Corporation

Pie charts

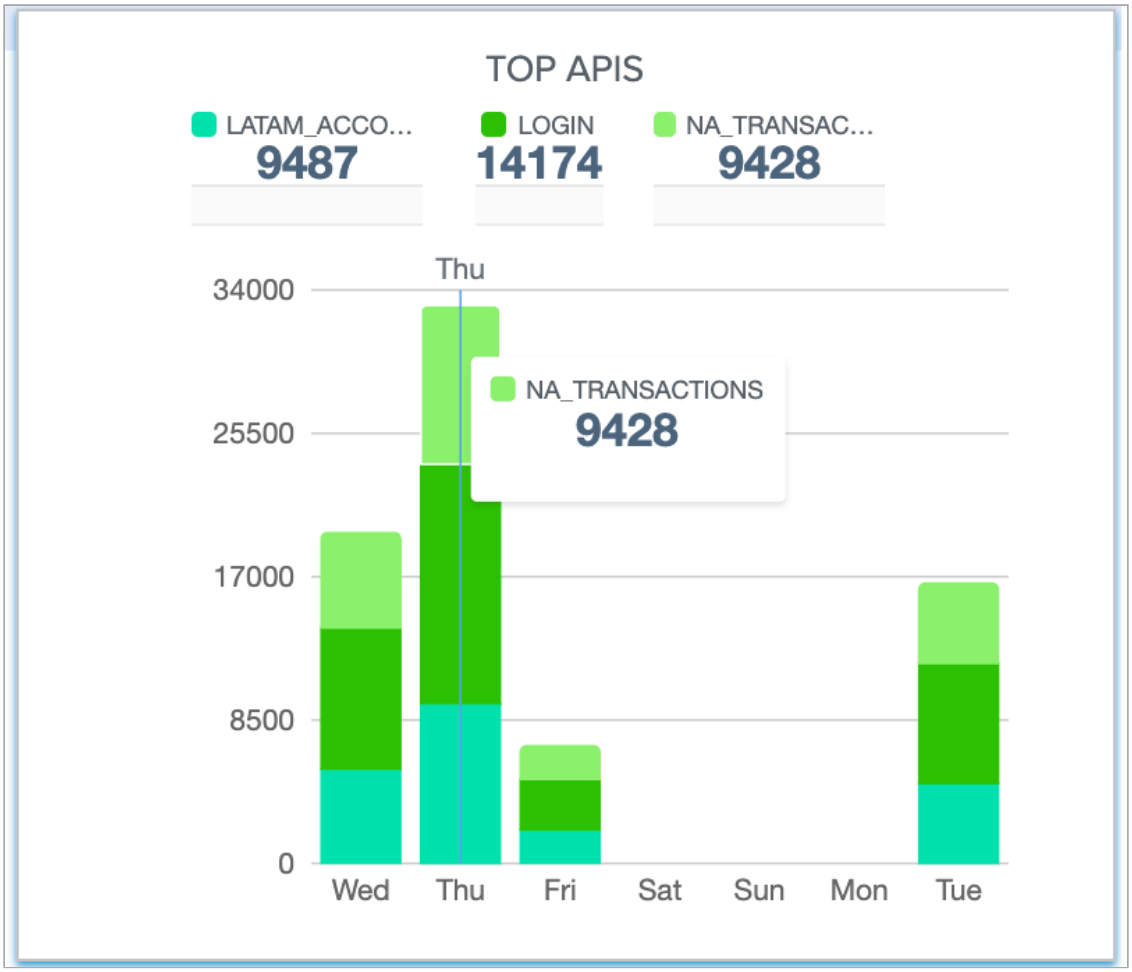
If you hover over a colored segment of the chart, the underlying value displays.



Bar charts

If you hover over a bar, you'll see a vertical line showing that it has focus and an underlying value. At the top, the values narrow down to those of the focused date.

Copyright © 2025 Ping Identity Corporation

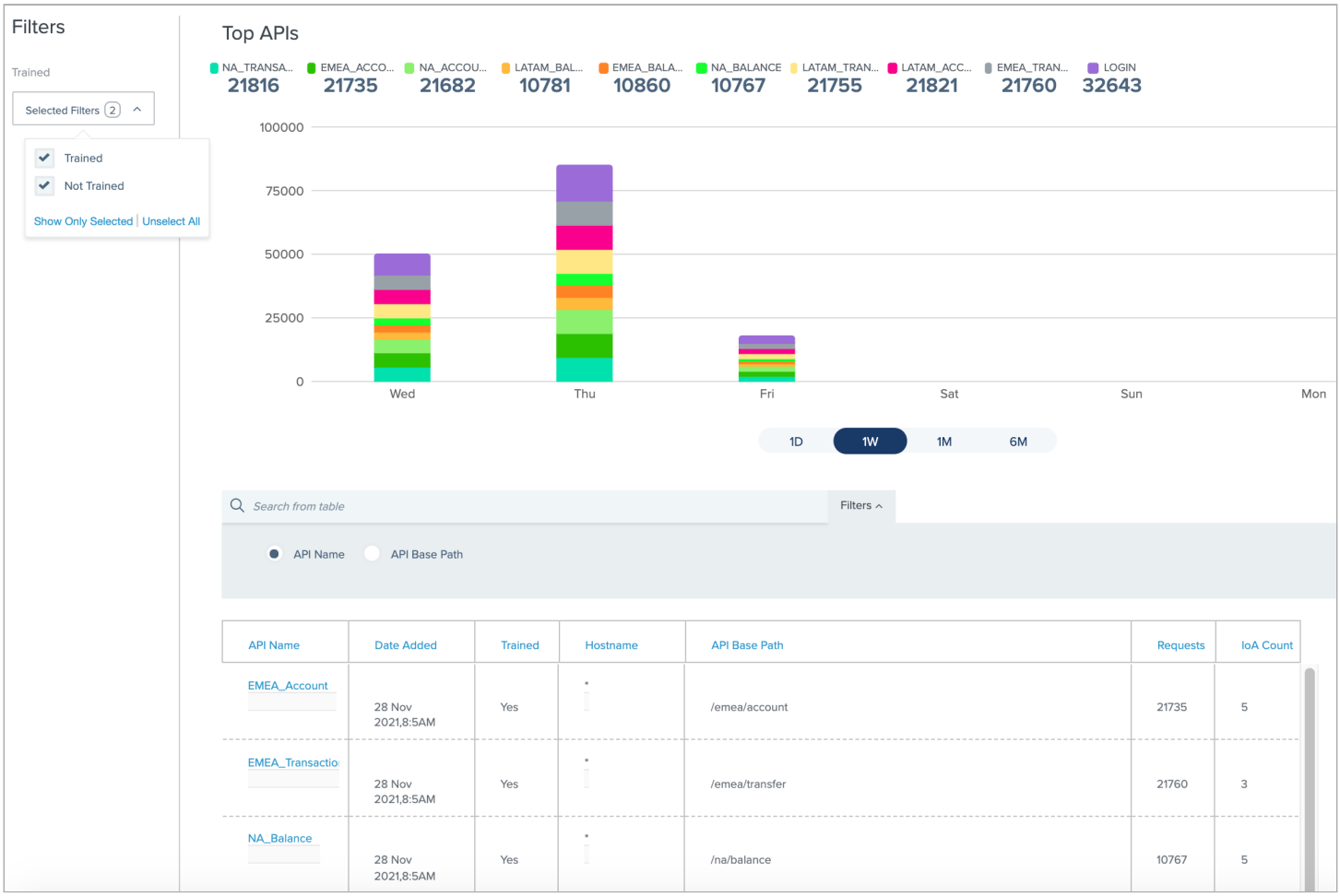


Top APIs

The Top APIs chart shows the distribution ratios of top application programming interface (API) request counts and indicators of attack (IoA) for the specified period.

To view the Top APIs chart details and drill-down information, click Dashboard → Top APIs.

Copyright © 2025 Ping Identity Corporation



Graph filters

The Trained graph filters are available. Select or clear the check boxes to include or exclude Trained or Not Trained APIs, for the specified selection.

Top APIs table

The filtered graph results are displayed in a table below the graph, for further drill-down, inspection and analysis.

Column	Description
API Name	The name of the API.
Date Added	The date monitoring began on the API.
Trained	Indicates whether the AI engine is trained (Yes) or not trained (No) on the API. **Note** An API must be trained before it can detect IoAs.
Hostname	The server hosting the API.

Copyright © 2025 Ping Identity Corporation

Column	Description
API Base Path	The Uniform Resource Identifier (URI) prefix for the API path, relative to the host root.
Requests	Accumulated count of requests to the API for the reported period.
IoA count	Accumulated count of IoAs on the API for the reported period.

Sorting and filtering

Click on a table column heading to sort the table according to that column. Click on the column heading again to display a reverse sort according to that column.

Click Filters ^ to select filtering based on search strings or partial strings of the API Name or API Base Path. Enter the search string in the Search from table field. The search displays the matching rows in the table.

> **Note**
>
> - The search is case-insensitive.
> - Wildcard searches, for example using an asterisk (`\*`), are not supported.
> - Use of quotation marks is not supported.
> - Be aware of the use of spaces in a search string. A leading or trailing space can filter out results. A single space is not regarded as multiple consecutive spaces.
> - The graph filters are applied before the table filter. Use of the table filter produces a subset of the graph filter. For example, in the graph filter select **Not Trained**. The graph and the table show not trained data only. The table's **Trained** column only has rows with the value `No` .

Select API Base Pathin the table filter, and enter `/acc` . The table is reduced further, displaying only the rows where the API Base Path column contains the string `/acc` .

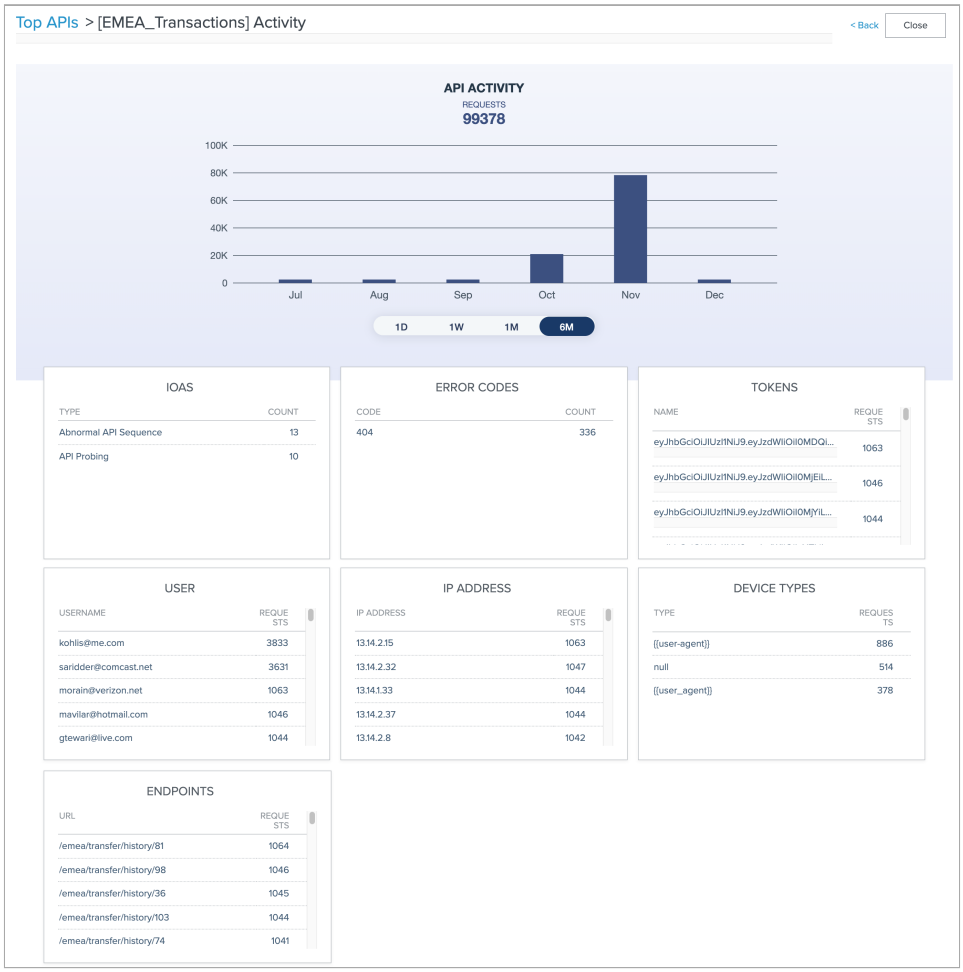
API drill-down

In the Top APIs table, click on an API in the API Name column, to display the API activity dashboard with further details about that API's activity in the reported period.

API activity

The API activity dashboard provides breakdown details about an application programming interface (API) activity in the reported period.

Copyright © 2025 Ping Identity Corporation



The API's request counts for the reported period display in the upper part of the API Activity dashboard. You can adjust the reporting period to display the API's request counts for the past day, week, month or 6 month period.

In the lower part of the screen the following panes display details about requests to the API, for the reported period.

Pane	Description
IoAs	**List of the types of IoAs detected on the API, and the count of IoAs per type.**
Error Codes	**List of error codes returned from requests to the API.**
Tokens	**List of the names of tokens used when issuing requests to the API, and the count of requests per token.**
User	**List of the usernames issuing requests to the API, and the count of requests per username.**
IP Address	**List of the Internet Protocol (IP) addresses where requests to the API originated, and the count of requests per IP address.**

Copyright © 2025 Ping Identity Corporation

Pane	Description
Device Types	List of the device types where requests to the API originated, and the count of requests per device type.
Endpoints	List of the API endpoints that received requests, and the count of requests per API endpoint.

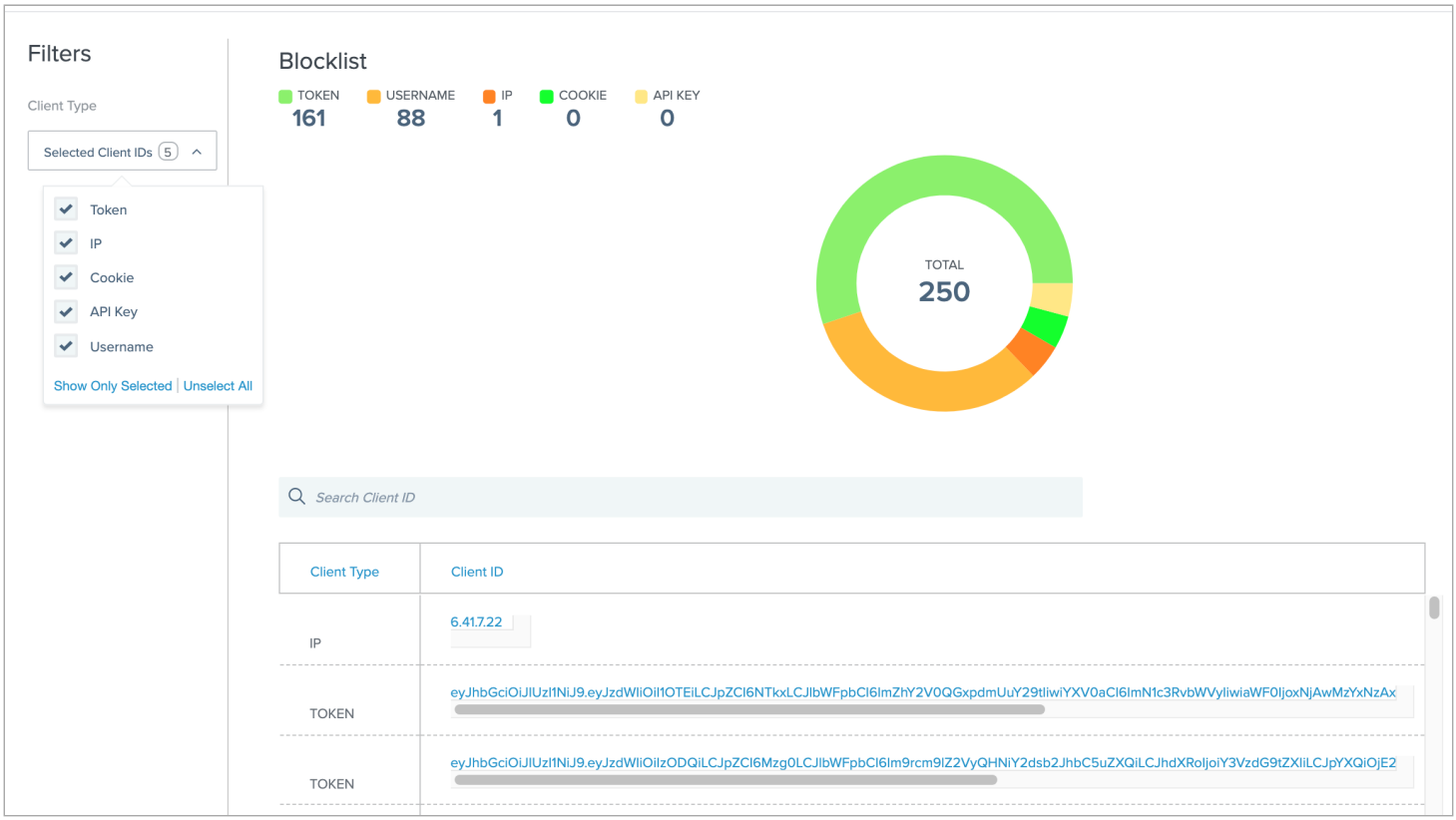
Click Back to return to the previous dashboard.

Click Close to return to the main dashboard screen.

Blocklist

The Blocklist chart shows the distribution of blocklists by client type, for the specified period.

To view the Blocklist chart details and drill-down information, click Dashboard → Blocklist.



Graph filters

The following Client type graph filters are available. Select or clear the check boxes to include or exclude blocked Client types in the report, for the specified period:

- Token
- IP

Copyright © 2025 Ping Identity Corporation

- Cookie
- API Key
- Username

Blocklist table

The filtered graph results are displayed in a table below the graph, for further drill-down, inspection and analysis.

Column	Description
Client type	The type of client.
Client ID	The unique ID of the client that is blocked.

Sorting

Click on a table column heading to sort the table according to that column. Click on the column heading again to display a reverse sort according to that column.

API drill-down

In the Clients table, click on an entry in the Client ID column, to display the Client activity dashboard with further details about that client's activity in the reported period.

Client activity

The Client activity dashboard provides breakdown details about a client's application programming interface (API) activity in the reported period.

You can navigate to the Client activity dashboard in one of the following ways:

- Go to Dashboard* → *Blocklist.

  In the blocked Clients table, click the entry in the Client ID column to navigate to its detailed breakdown in the Client activity dashboard.

- Go to Attack management.

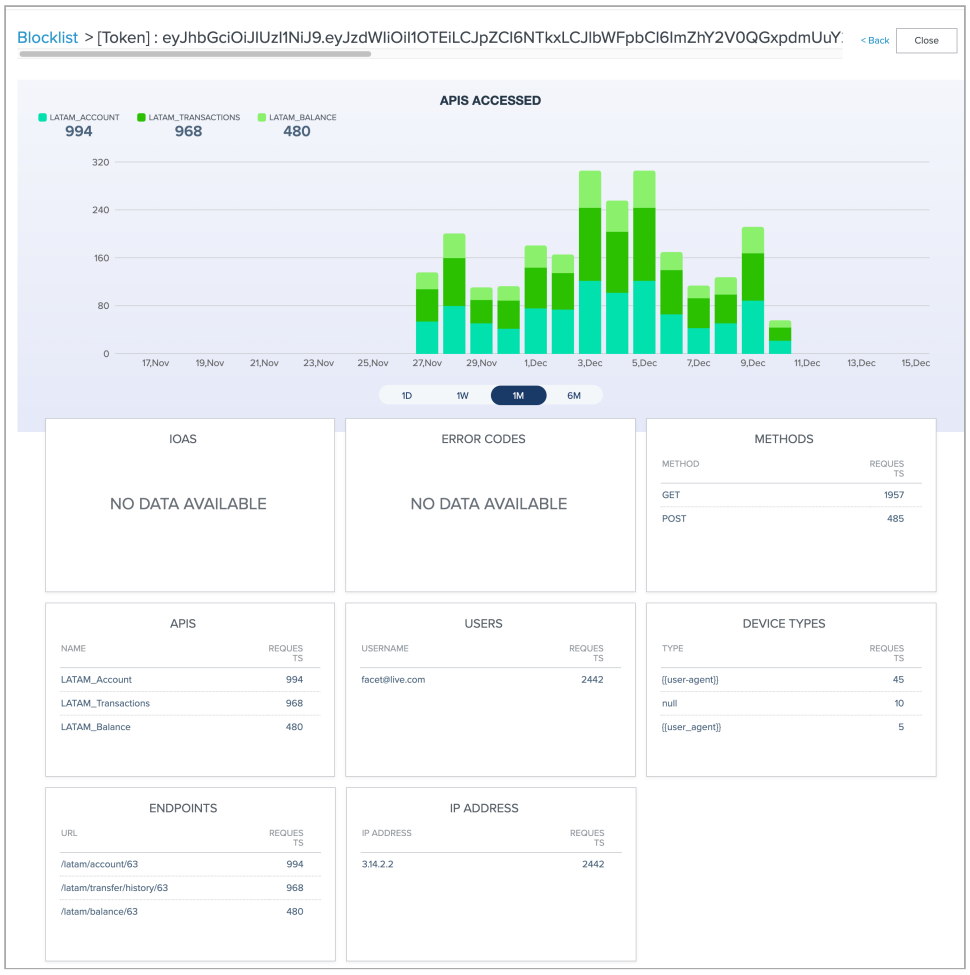
  Choose:

  - On the right side of a client's row in the main Attack management list, click the three-dots drop down, then click Client activity to navigate to its detailed breakdown in the Client activity dashboard.
  - Click a client's row in the main Attack management list to navigate to the client's IoAs dashboard. At the top of the screen, on the right side of the client's summary line, click the three-dots drop down, then click Client activity to navigate to its detailed breakdown in the Client activity dashboard.

At the top right, Close returns you to the previous dashboard screen that you viewed.

Copyright © 2025 Ping Identity Corporation

If you navigate to the Client activity dashboard from the Blocklist dashboard, the Blocklist and Back navigation controls also appear at the top of the screen, and will return you to the Blocklist dashboard.



The summary count of APIs accessed by the client for the reported period display in the upper part of the Client activity dashboard. You can adjust the reporting period to display counts of APIs accessed for the past day, week, month or six-month periods.

In the lower part of the screen, the following panes display details about the APIs accessed by the client, for the reported period:

Pane	Description
IoAs	List of the types of IoAs based on the client's activity, and the count of IoAs per type.
Error Codes	List of error codes returned from the client's requests to the APIs.
Methods	List of the API methods used when the client issued API requests, and the count per method.

Copyright © 2025 Ping Identity Corporation

Pane	Description
APIs	List of the APIs receiving requests from the client, and the count of requests per API.
User	List of the usernames issuing requests to the API, and the count of requests per username.
Device Types	List of the device types where requests to the API originated, and the count of requests per device type.
Endpoints	List of the API endpoints that received requests, and the count of requests per API endpoint.
IP Address	List of the Internet Protocol (IP) addresses where requests to the API originated, and the count of requests per IP address.

Click Close at the top right, to return to the previous dashboard screen.

Client IoAs

The Client IoAs (Indicators of Attack) summary chart on the main dashboard screen lists the top IoA counts per client, in descending order.

Copyright © 2025 Ping Identity Corporation

Click Dashboard → Client IoAs to view the Attack list on the Attack management dashboard for further drill-down, inspection and analysis.

APIs

API groups

The PingIntelligence dashboard provides the capability to organize the application programming interface (API) in your environment into logical groups. You can create API groups as per your requirements. For example, you can group your APIs location-wise, functionality-wise, and so on.

The API tab displays all the APIs being managed by PingIntelligence. Every API will be part of at least one API group in the Dashboard. The APIs grouping feature makes searching for a specific API quick and easy. The Dashboard supports two kinds of groups:

- Default API group: This is the global API group. All the existing as well as newly discovered APIs will be part of it initially. APIs that do not belong to any other API group will automatically get added to the default API group. You can only view and move APIs from the default APIs group. You cannot delete an API from the default API group.

Copyright © 2025 Ping Identity Corporation

- User-defined API groups: These are the API groups that you can create based on your requirements. You can add or delete an API from the user-defined API groups.

API details

Click on the Expand icon  to expand an API group. The following details are available for each API within an API group.

API	Description
API name	API name used by PingIntelligence
Prediction mode	A `true` status means that at least one training threshold value is set. It does not necessarily mean that all the training is complete. A `false` status means that the API is still in training mode.
Training duration	The minimum configured time in hours to train an API. For more information, see Managing AI engine training.
URL	API basepath Uniform Resource Locator (URL) configured in the API JavaScript Object Notation (JSON) file. For more information, see Defining an API using API JSON configuration file in inline mode.
Host name	Host name of the API configured in the API JSON file. For more information, see Defining an API using API JSON configuration file in inline mode.
Protocol	The protocol configured in the API JSON file. For more information, see Defining an API using API JSON configuration file in inline mode.

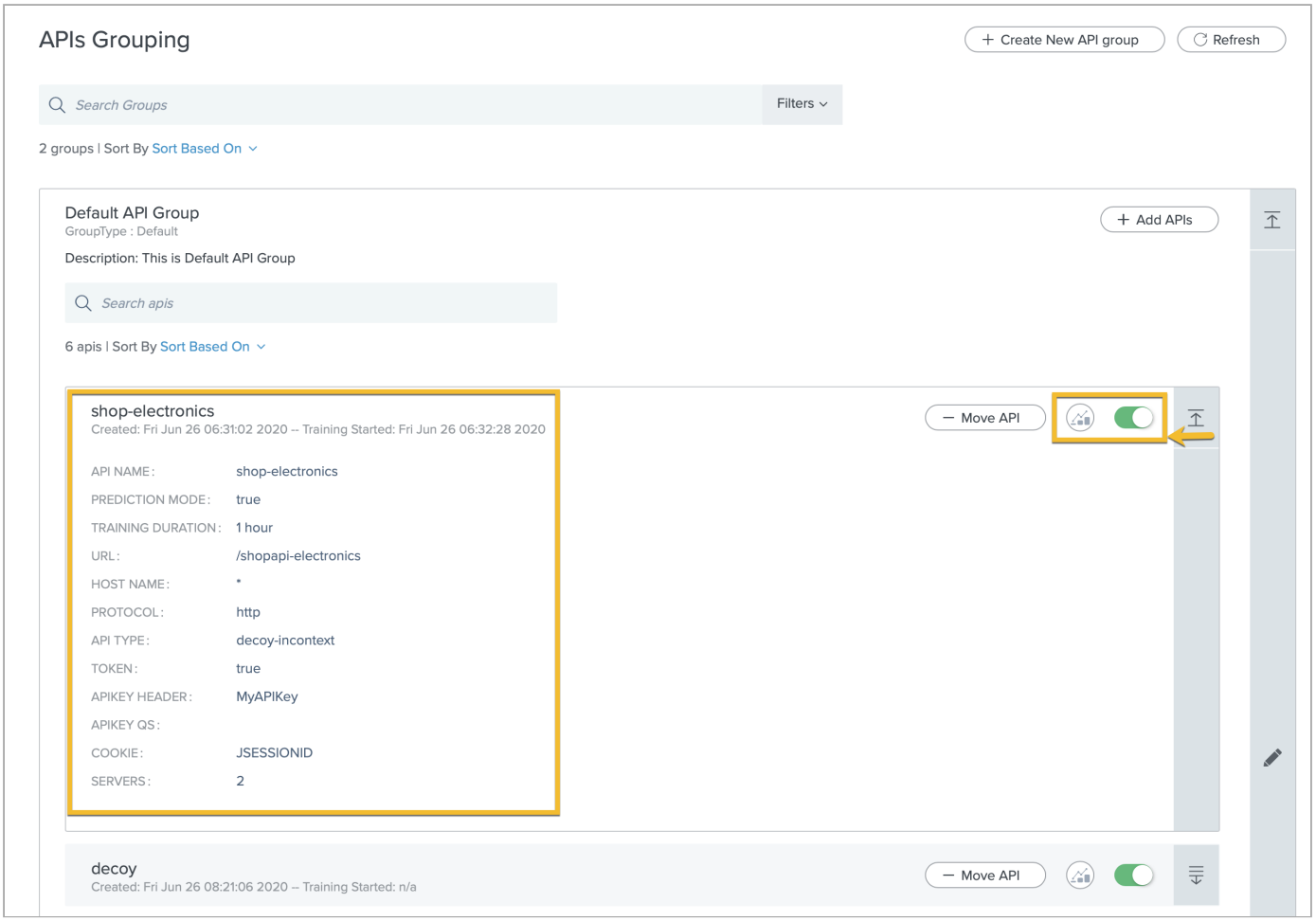
Copyright © 2025 Ping Identity Corporation

API	Description
API type	API type can be `regular`, `decoy - incontext`, or `decoy-out-of-context`. For more information on deception, see API deception environment in inline mode.
Token	A `true` status means that PingIntelligence will use OAuth tokens for reporting and attack detection. For more information, see Defining an API using API JSON configuration file in inline mode.
API Key header and API key query string (QS)	The API Key values configured in the API JSON file and used for reporting and attack detection. For more information, see Defining an API using API JSON configuration file in inline mode.
Cookie	The cookie value configured in the API JSON file and used for reporting and attack detection. Displays blank if a cookie was not configured in API JSON. For more information, see Defining an API using API JSON configuration file in inline mode.
Servers	The back end API server configured in the API JSON file - "*" supports all the host names. For more information, see Defining an API using API JSON configuration file in inline mode.

Click the Toggle button  to hide or display information for the API in the PingIntelligence Dashboard. This provides the flexibility to display only selected APIs. Even if an API is hidden from the API dashboard, the dashboard engine continues processing its metadata. The hidden API is moved to the end of list. If the APIs are paginated, the hidden APIs are moved to the last page. When you toggle the button to display a hidden API, the Dashboard displays data for the API on the Dashboard.

You can also go to the API activity dashboard for the API, by clicking the API analytics icon  .

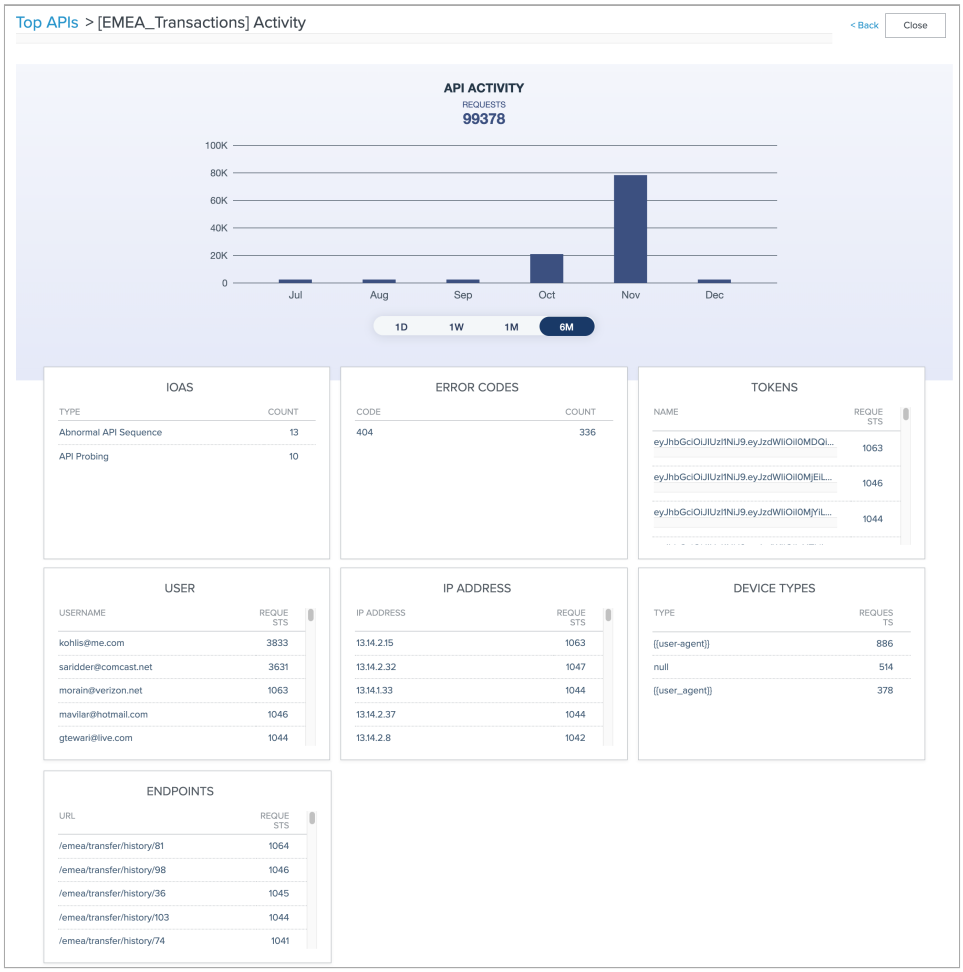
Copyright © 2025 Ping Identity Corporation



API activity

The API activity dashboard provides breakdown details about an application programming interface (API) activity in the reported period.

Copyright © 2025 Ping Identity Corporation



The API's request counts for the reported period display in the upper part of the API Activity dashboard. You can adjust the reporting period to display the API's request counts for the past day, week, month or 6 month period.

In the lower part of the screen the following panes display details about requests to the API, for the reported period.

Pane	Description
IoAs	List of the types of IoAs detected on the API, and the count of IoAs per type.
Error Codes	List of error codes returned from requests to the API.
Tokens	List of the names of tokens used when issuing requests to the API, and the count of requests per token.
User	List of the usernames issuing requests to the API, and the count of requests per username.
IP Address	List of the Internet Protocol (IP) addresses where requests to the API originated, and the count of requests per IP address.

Copyright © 2025 Ping Identity Corporation

Pane	Description
Device Types	List of the device types where requests to the API originated, and the count of requests per device type.
Endpoints	List of the API endpoints that received requests, and the count of requests per API endpoint.

Click Back to return to the previous dashboard.

Click Close to return to the main dashboard screen.

Administering API groups

This topic discusses administrative tasks associated with your application programming interface (API) groups.

Before you begin

Make sure you have admin user privileges to administer the API groups in the dashboard.

About this task

You can perform the following administrative operations on API groups:

- Creating and deleting API groups
- Adding, deleting, and moving APIs
- Merging a user-defined API group into the default API group
- Searching or sorting API groups and APIs

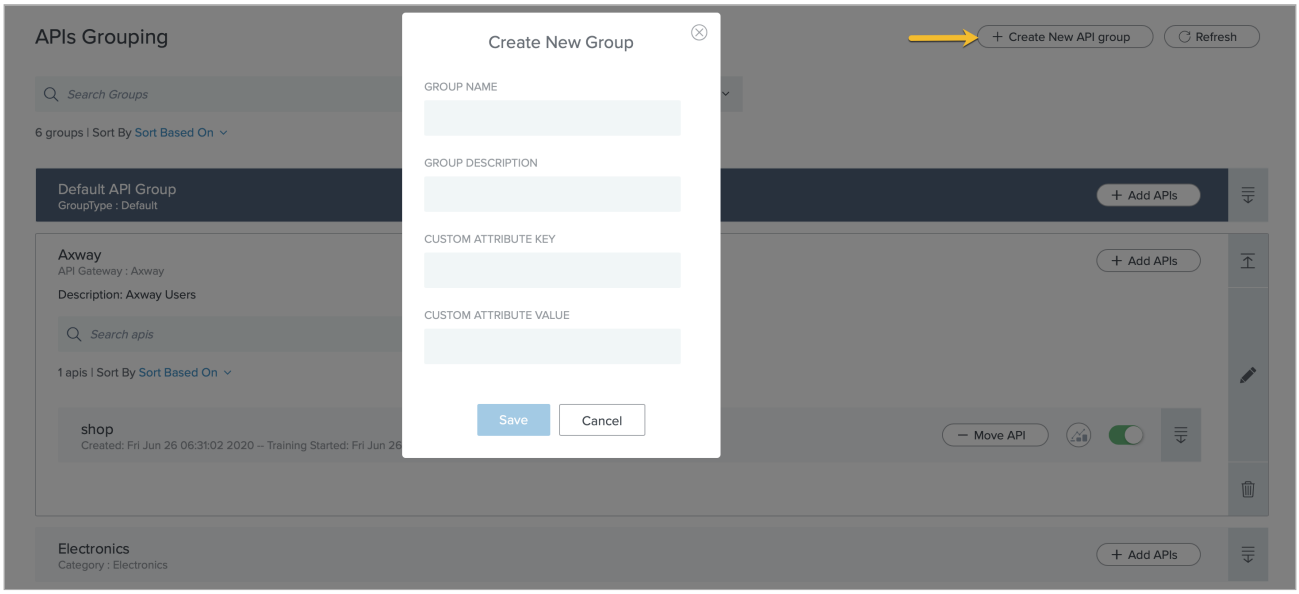
> **Note**
>
> A successful execution of these operations is followed by a success notification. Click the **Refresh** 
>
> button on the top-right corner, to reflect the changes made to the API groups.

Creating and deleting API groups

Steps

- To create an API group, click Create New API group on the top-right corner. Fill in the following details for the new API group, and click Save:
  - Group name: The display name of the API group.
  - Group description: Additional information about the API group.
  - Custom attribute key: The metadata key for the API group.
  - Custom attribute value: Metadata about the API group. This can be used in search operations.

Copyright © 2025 Ping Identity Corporation



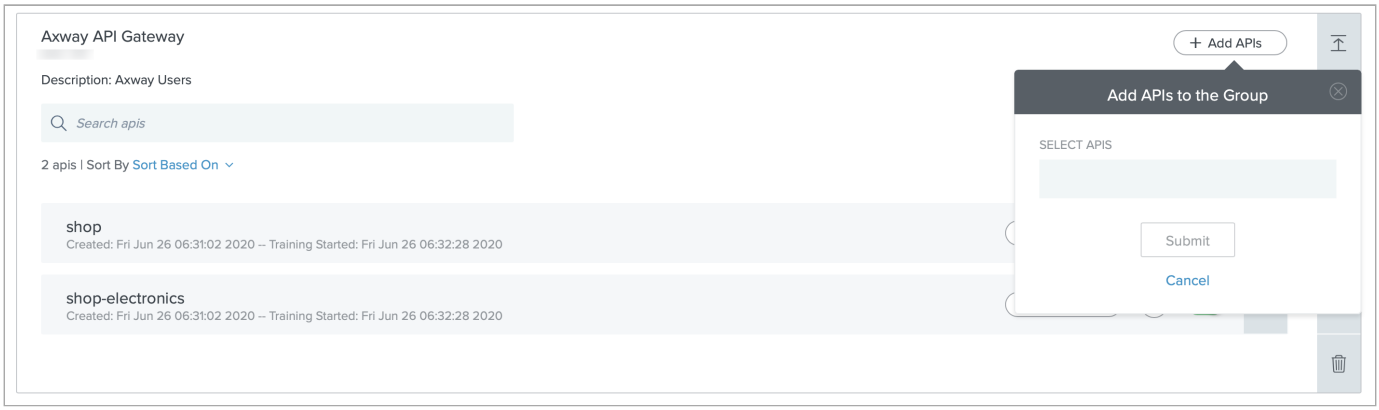
- Click the Pencil  icon, to edit an API group. You can modify the metadata of the group.
- Click the Delete  icon on the bottom-right corner of the API group, to delete an API group. APIs in the group that are not part of any other API groups, will be added to the default API group. You cannot delete the default API group.

Adding, deleting, and moving APIs

You can add, delete, or move an API from an API group.

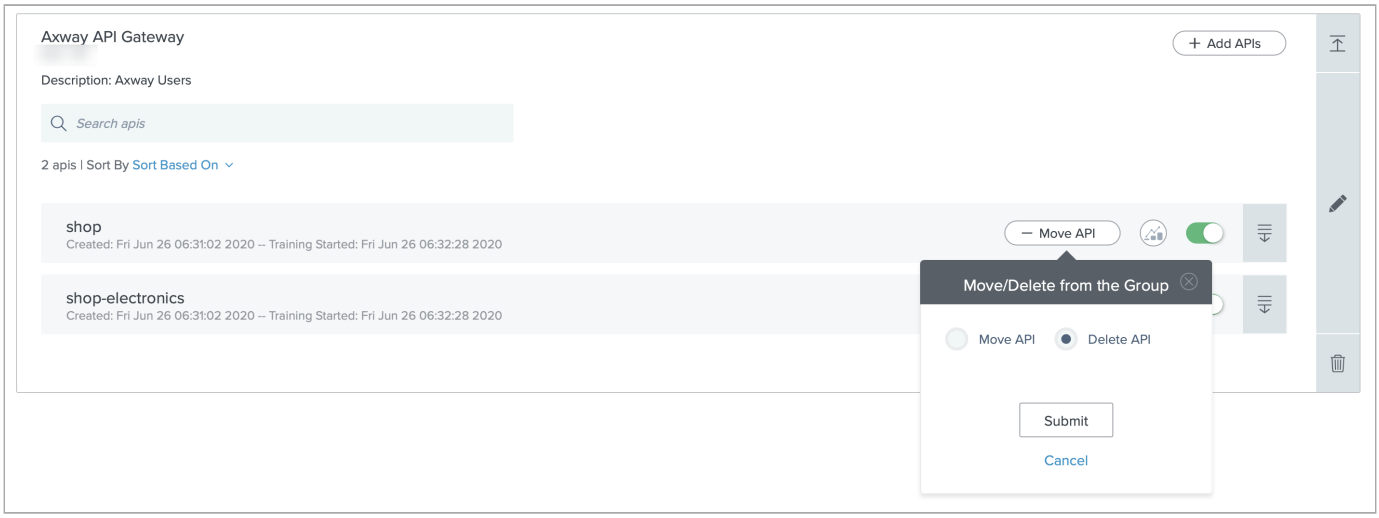
Steps

- To add an API to group, click Add APIs on the top-right corner of the API group. Select the API from the Add APIs to the Group pop-up and click Submit. You can select more than one API and add them to a group in one instance.

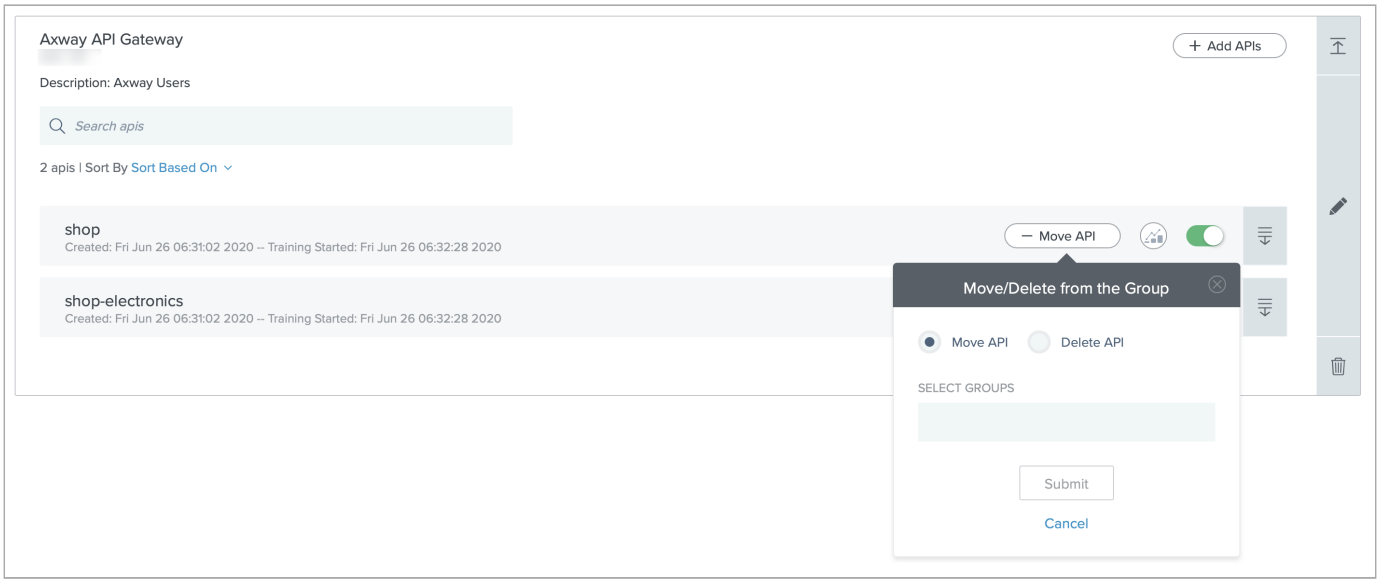


- To delete an API from an API group, click Move API. Select Delete API in the Move/Delete from the Group pop-up. Click Submit. After an API which does not belong to any other group is deleted from a group, then it automatically gets added to the default group.

Copyright © 2025 Ping Identity Corporation



- To move an API to a different API group, click Move API. Select Move API in the Move/Delete from the Group pop-up. Now select the target API group and click Submit. You can move the API to more than one target API groups. Once the API is moved it'll no longer be part of that API group.



Merging a user-defined API group into the default API group

You can merge a user-defined API group into the default API group.

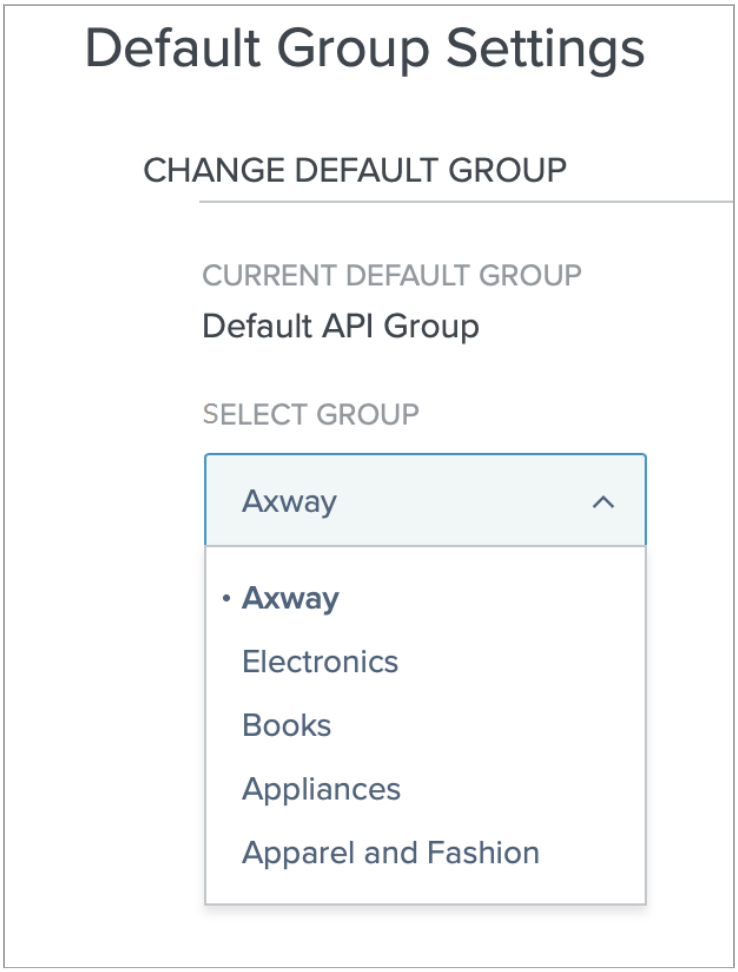
About this task

Complete the following steps:

Steps

1. Click Settings → API grouping settings.
2. From the Select group list, select the API group that you want to move to the default group.
3. Click Save.

Copyright © 2025 Ping Identity Corporation



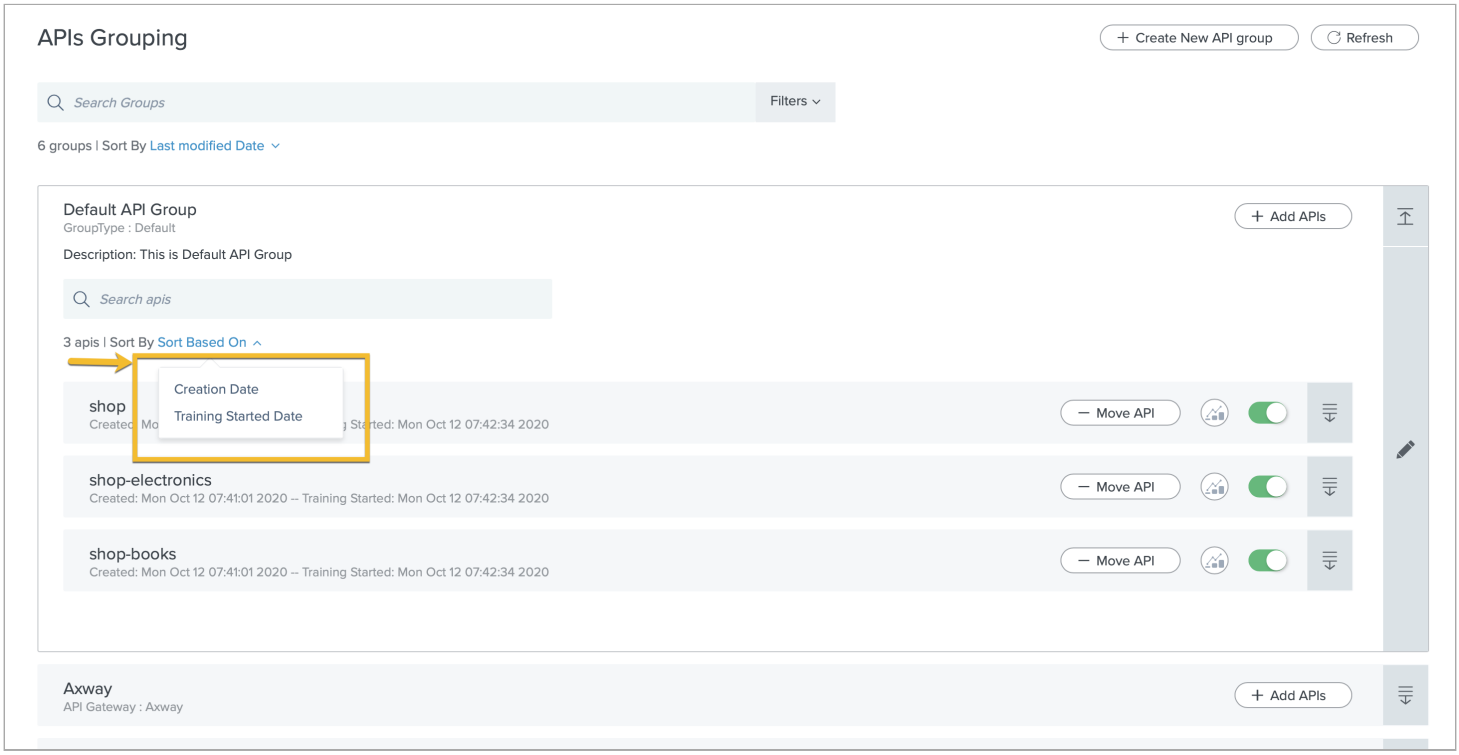
Searching or sorting API groups and APIs

Steps

- **You can search for a specific API in an API group or across multiple API groups. For quick and easy retrieval, when you search at the API group level, you can filter your search based on Group name, Attribute, or API. When API is chosen for filtering, only non-empty API groups are loaded.**
- **You can sort API groups based onGroup name, Group creation date, or Last modified date.**

Copyright © 2025 Ping Identity Corporation

- **You can also sort the APIs within an API group based onCreation date or Training start date.**



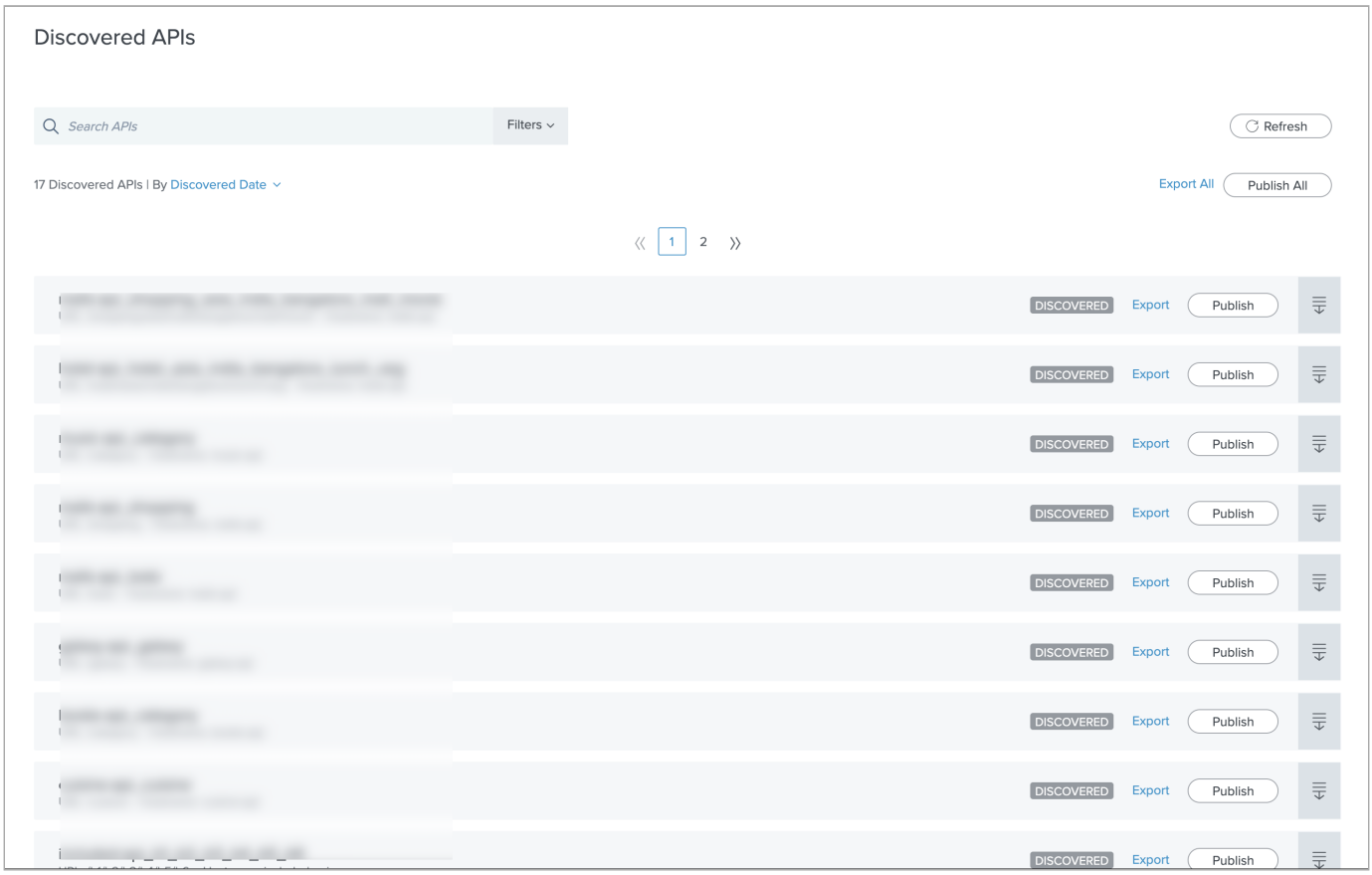
Discovered APIs

API discovery is a process to discover application programming interface (API) in your API environment.

Copyright © 2025 Ping Identity Corporation

The discovery process involves all PingIntelligence components:

- API Security Enforcer (ASE) - A `root` API is defined in ASE for the discovery process to start. The `root` API access log data is sent to API Behavioral Security (ABS) AI engine for processing.
- ABS AI engine - The ASE access logs are processed to discover APIs in your environment.
- Dashboard - Displays, manages, and renders the discovered APIs. Dashboard allows you to edit the discovered APIs and publish them to ASE. To view the APIs discovered from your API ecosystem, navigate to Discovered APIs in the Dashboard as shown in the screenshot below.



Configure API discovery

To customize the discovery process, configure the discovery parameters on the Dashboard.

Navigate to Settings → Discovered API.

Discovery settings consists of the following three parts:

- Mode - Configure the mode in which application programming interface (API) are published to API Security Enforcer (ASE). The mode can be Manual or Auto.
- Discovery Configuration - Switch discovery on or off, configure the subpath depth of the API base path and discovery interval.

Copyright © 2025 Ping Identity Corporation

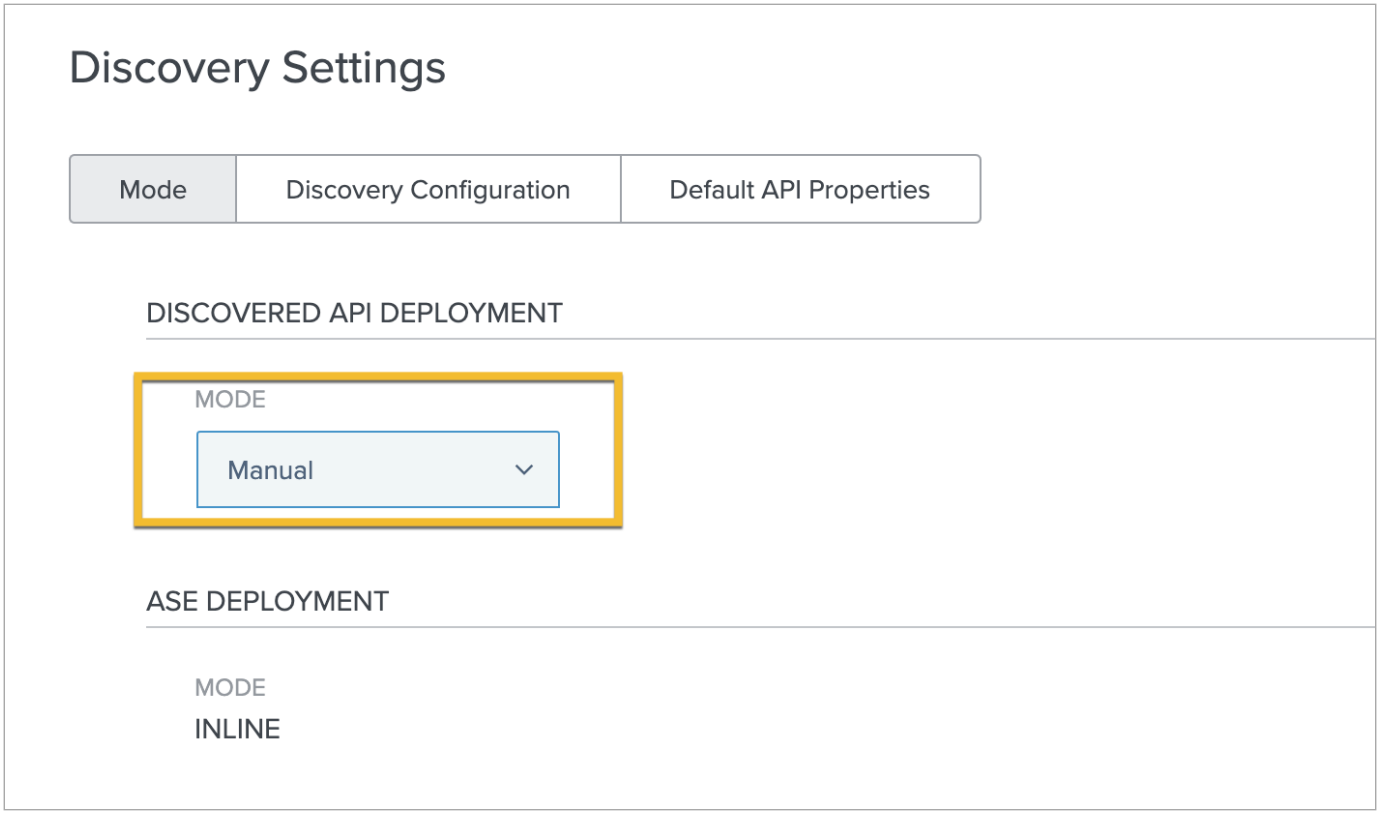
- Default API Properties - Configure the default properties of discovered APIs. You can edit the properties of an individual API in manual mode before publishing it to ASE.

The following sections explain each part of Discovery settings in detail.

Mode

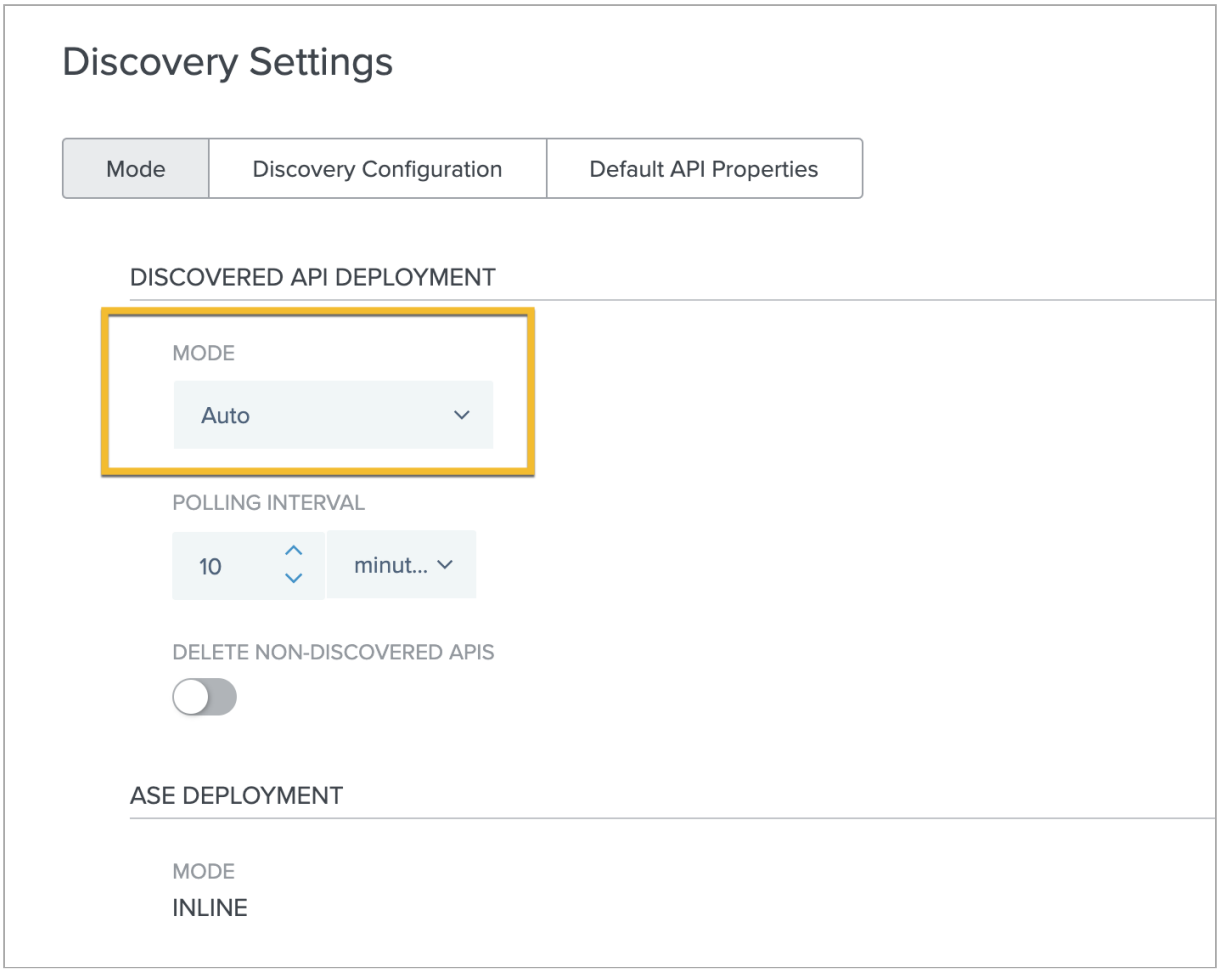
Configure the mode in which dashboard publishes the discovered APIs to ASE. The two modes are:

- Manual: In manual mode, you can review the discovered APIs, edit the properties of the APIs and then publish one or more APIs. For more information on editing the discovered APIs, see Editing discovered APIs.



- Auto: In auto mode, dashboard automatically publishes the APIs after a configured time interval. In auto mode, if you edit an API, it is published in the subsequent interval. Configure the following for auto mode:
  - Polling Interval - The time interval at which dashboard publishes APIs to ASE. It is a good practice to have a minimum of a 10 minute interval.
  - Delete non-discovered APIs - When enabled, any APIs manually added to ASE are deleted.

Copyright © 2025 Ping Identity Corporation



- ASE Deployment - Displays the ASE deployment mode - inline or sideband.

Discovery Configuration

Enable or disable discovery from the Discovery Configuration tab by toggling the AI Engine Discovery button. Configure the following:

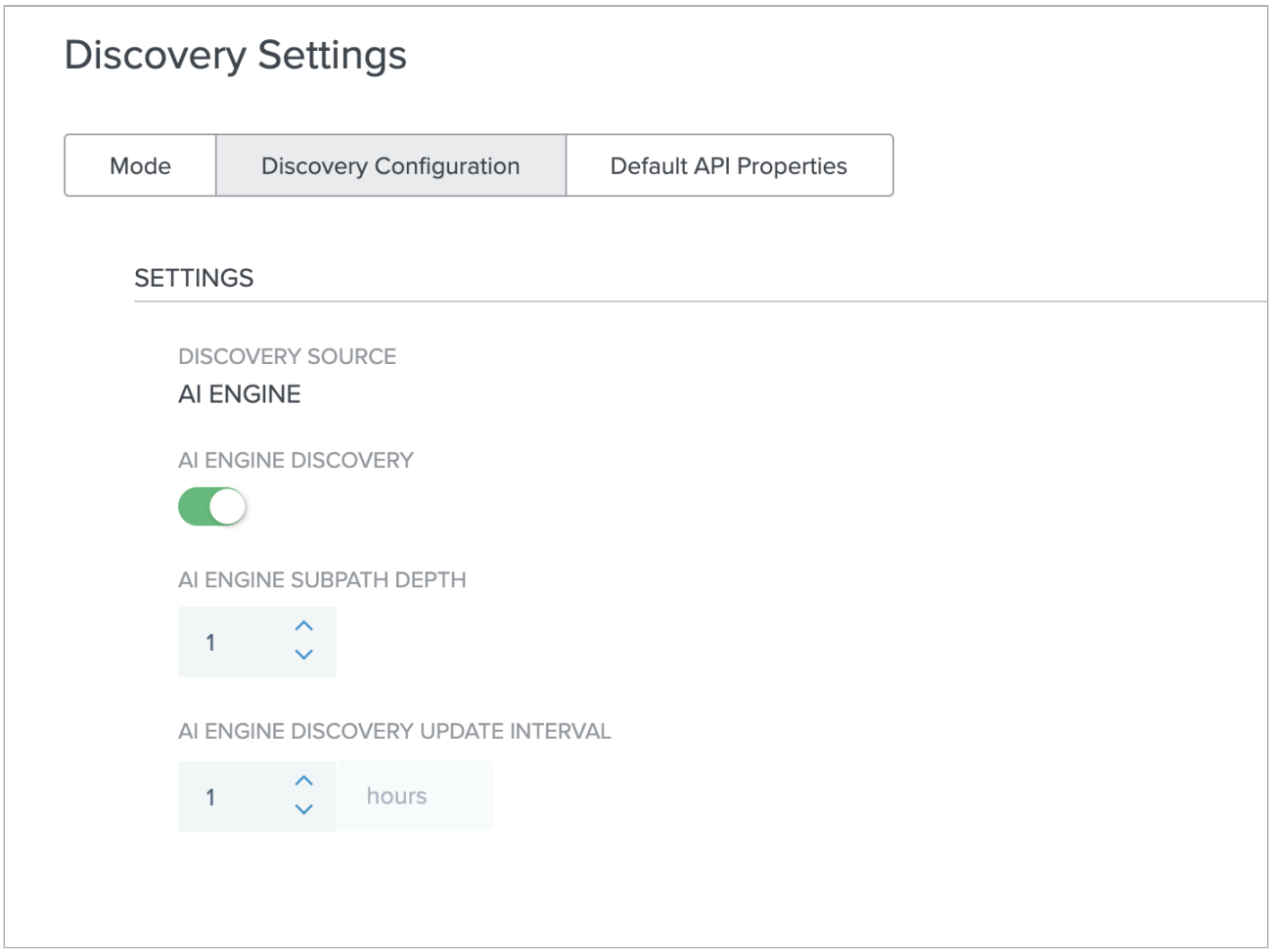
- Discovery Source - the source for newly discovered APIs. Different options are available based on the platform:
  - PingIntelligence for APIs software supports three sources:
    - AI engine
    - PingAccess
    - Axway API gateway

Refer to Configure API discovery for setup instructions.

- AI Engine Discovery - Toggle the button to start or stop API discovery. Make sure a root API is configured in ASE for the AI engine to discover APIs. For more information on discovery process, see API discovery and configuration.
- AI Engine Subpath Depth - Defines the number of subpaths used to uniquely discover the base path of a new API. The maximum allowed value is 6 when ASE is deployed in inline mode and it is 10 when ASE is deployed in sideband mode. For more information, see Discovery sub-paths.

Copyright © 2025 Ping Identity Corporation

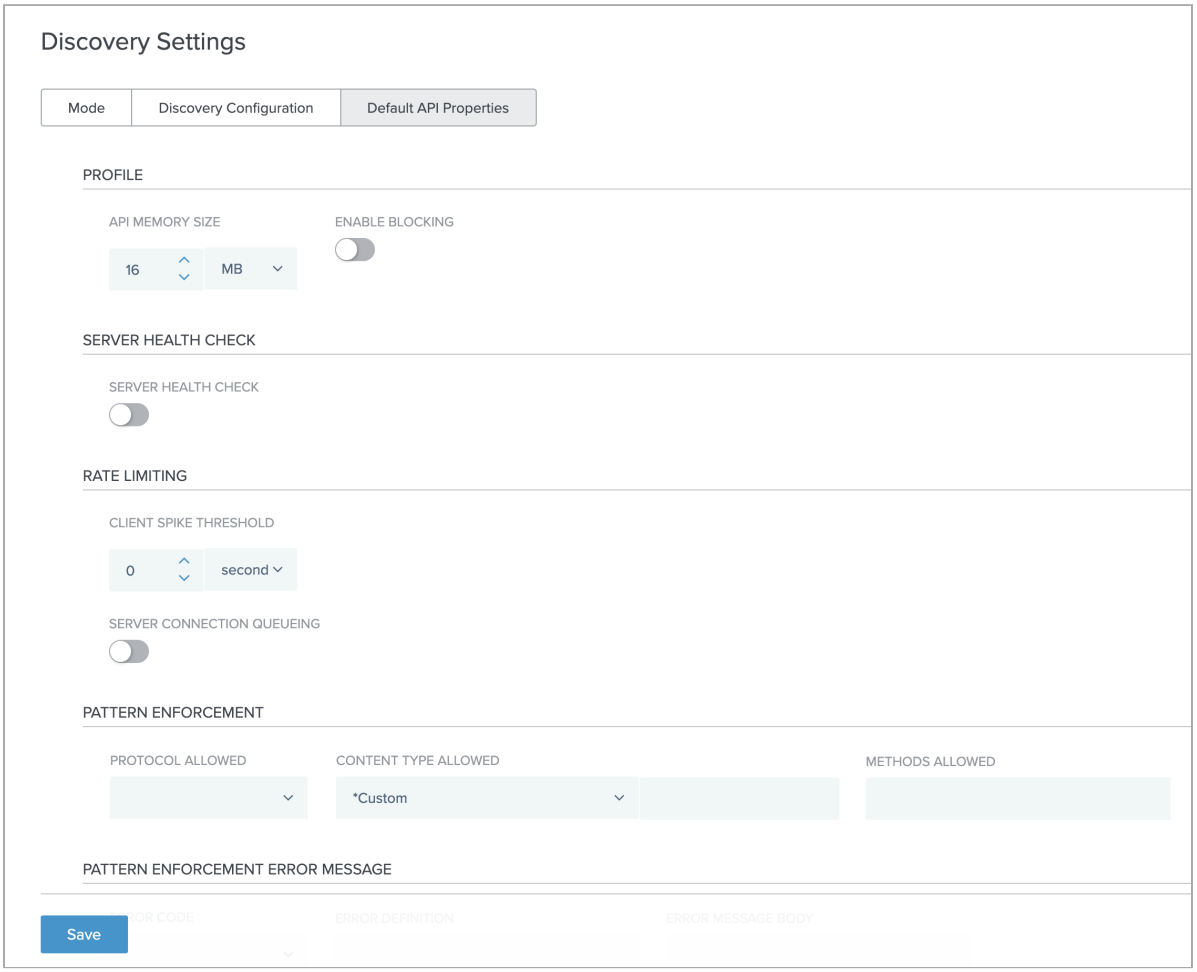
- **AI Engine Discovery Update Interval - Defines the time interval at which new discovered APIs are updated in the Dashboard. The minimum value is 1 hour.**



Default API Properties

You can configure the default API JavaScript Object Notation (JSON) properties from this tab. These properties apply to all discovered APIs. You can edit the properties of the discovered APIs in manual mode before publishing. For more information on the API properties, see Defining an API using API JSON configuration file in inline mode**.**

Copyright © 2025 Ping Identity Corporation



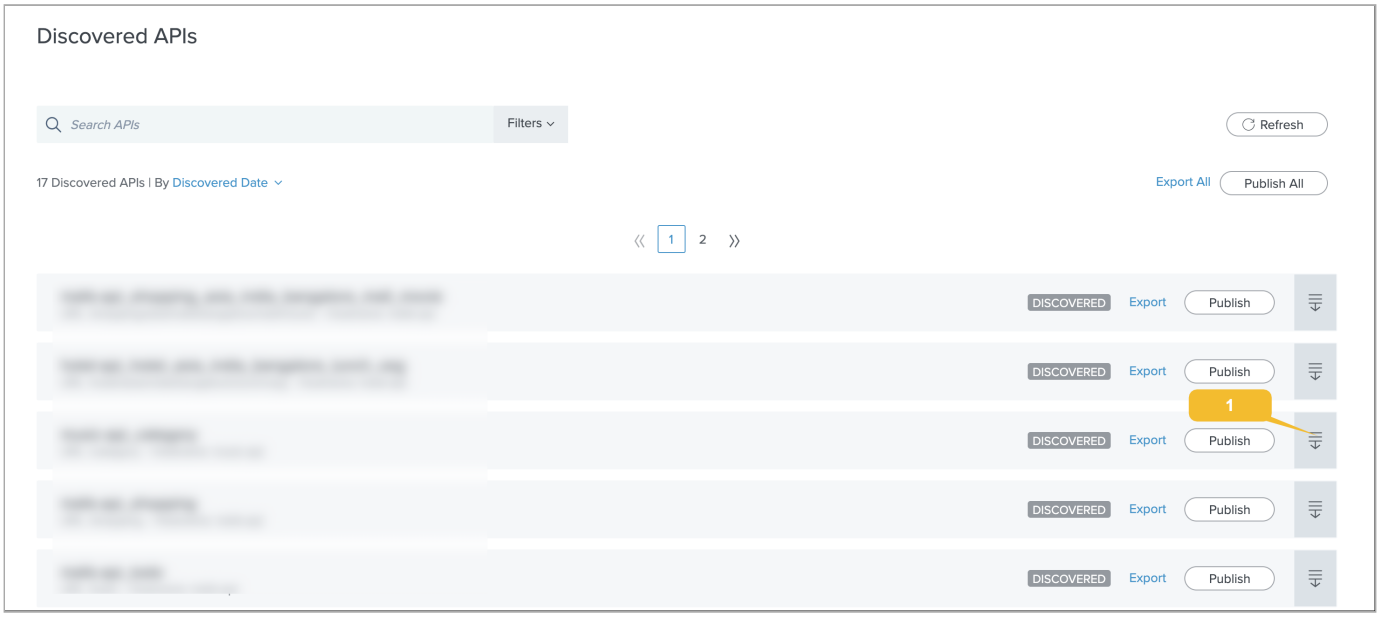
Editing discovered APIs

You can edit the discovered application programming interface (API) from the Discovered APIs page.

Steps

- To edit an API, click on the drop down arrow shown in the screen capture.

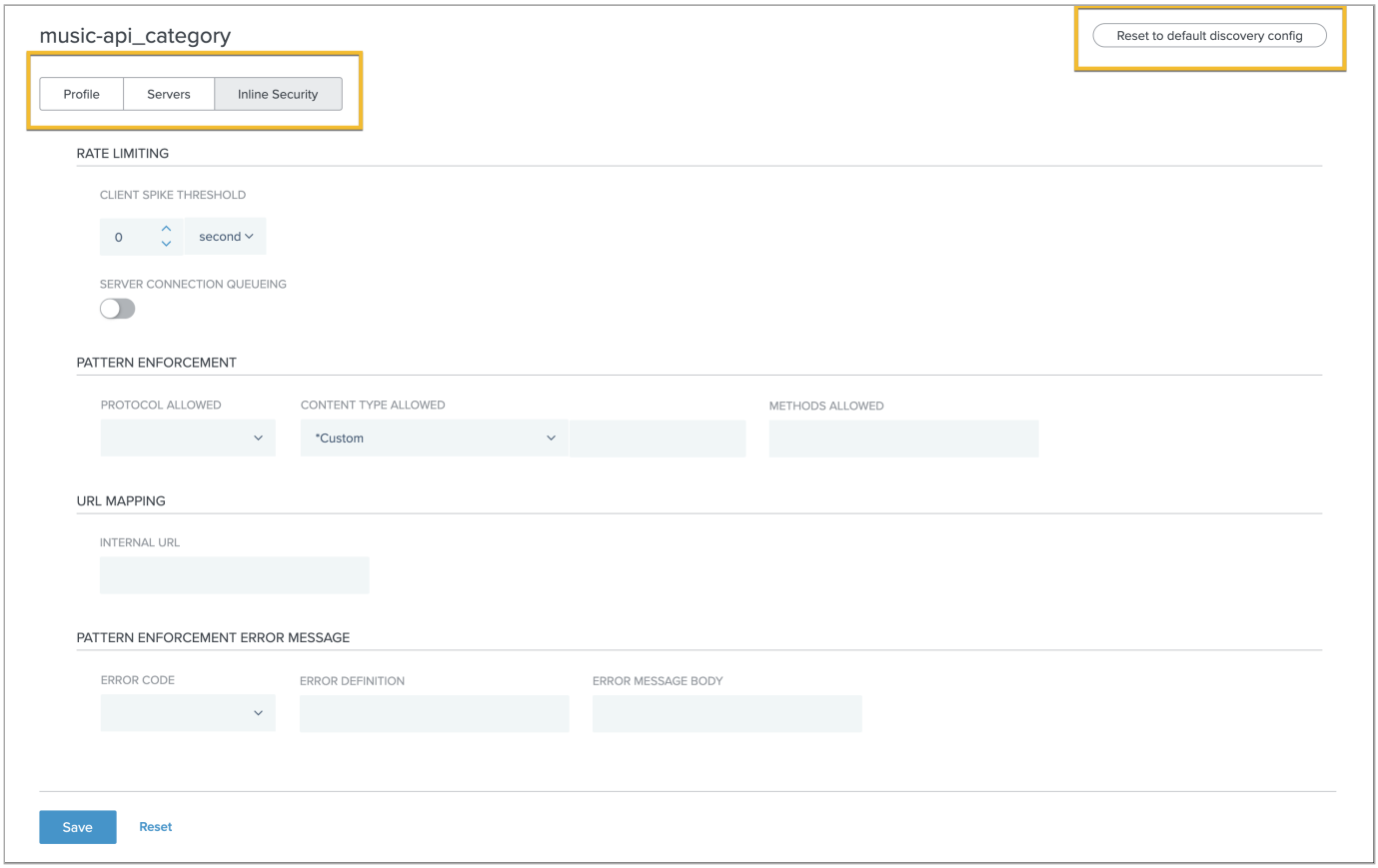
Copyright © 2025 Ping Identity Corporation



Note

You can download the API definition in JSON format by clicking on **Export**. Click **Publish** to add API to PingIntelligence and begin the training process. You will be notified on successful publication of the API.

The edit API page is displayed when you click on the drop down arrow.

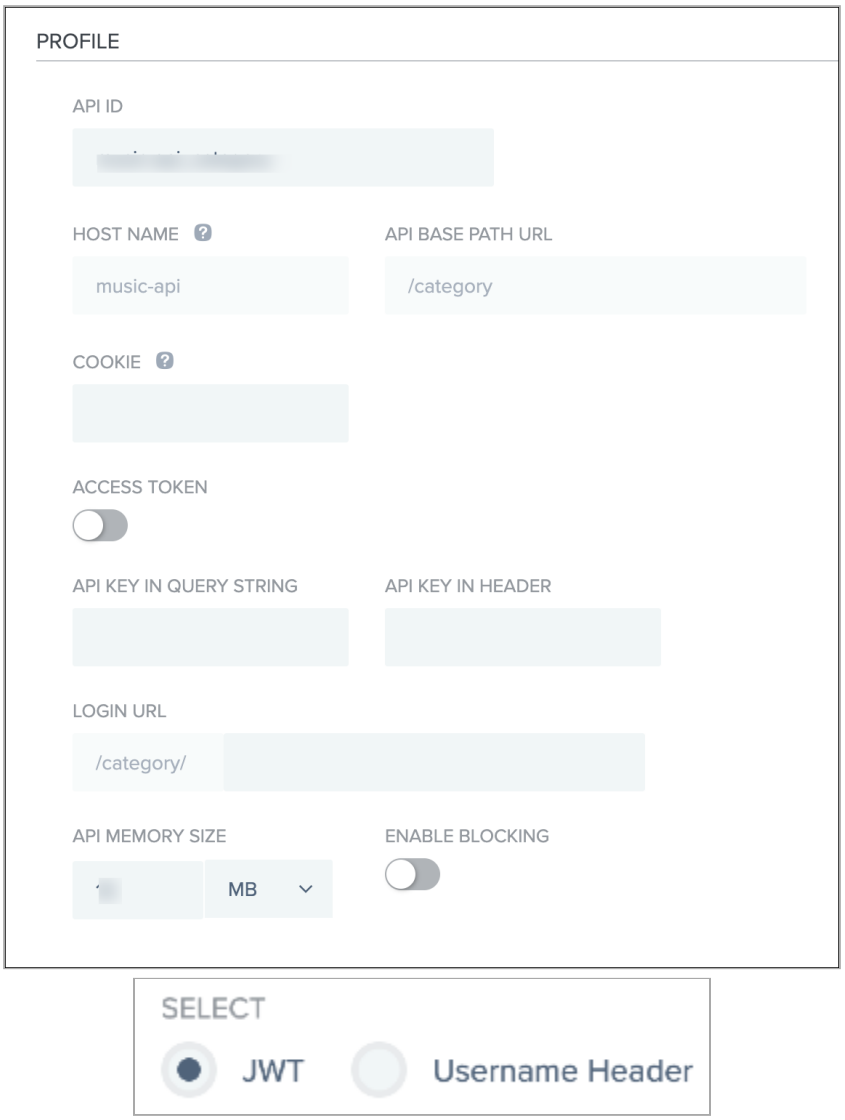


Copyright © 2025 Ping Identity Corporation

The edit API page allows you to set properties of an API JSON file. These are the same properties that you configure when you define an API JSON in API Security Enforcer (ASE). For more information on defining an API JSON, see Defining an API using API JSON configuration file in inline mode.

You can also reset the edited changes by clicking on the Reset to default discovery configure button on the top-right corner. This resets the API properties to the one that was set during the Configure API discovery step. The edit API page is divided into three tabs:

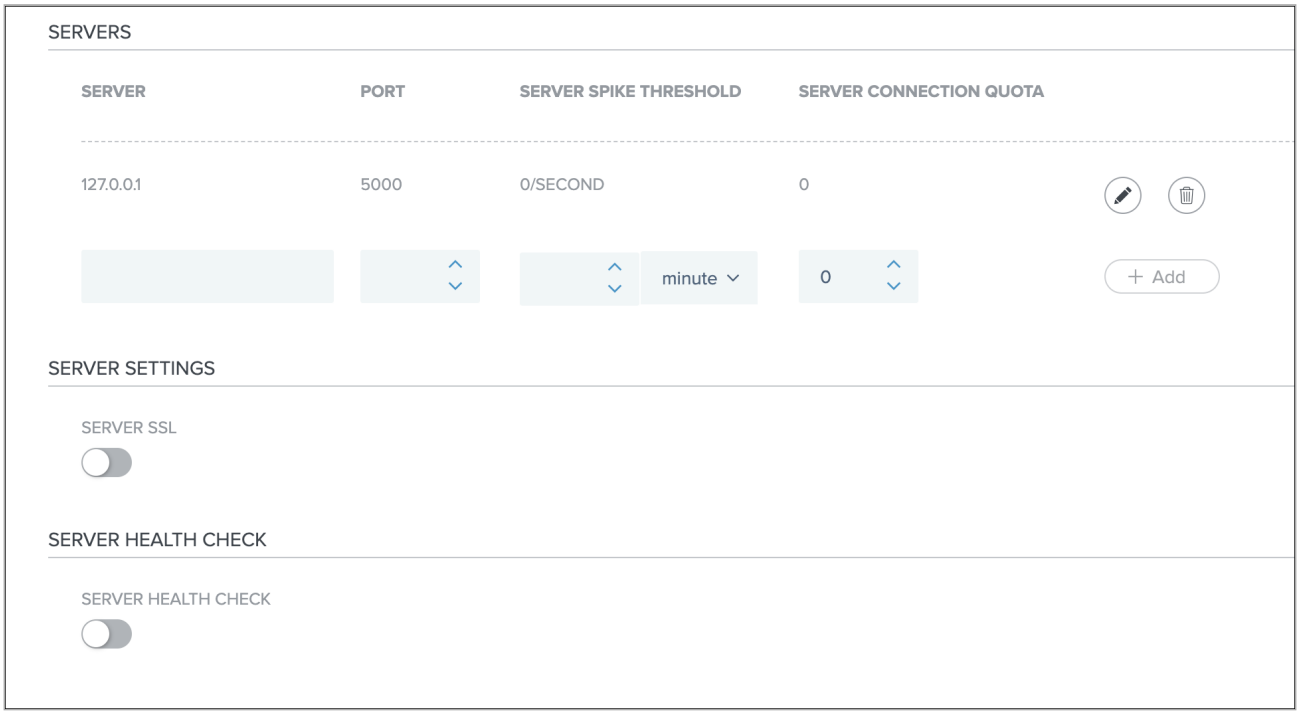
- Profile



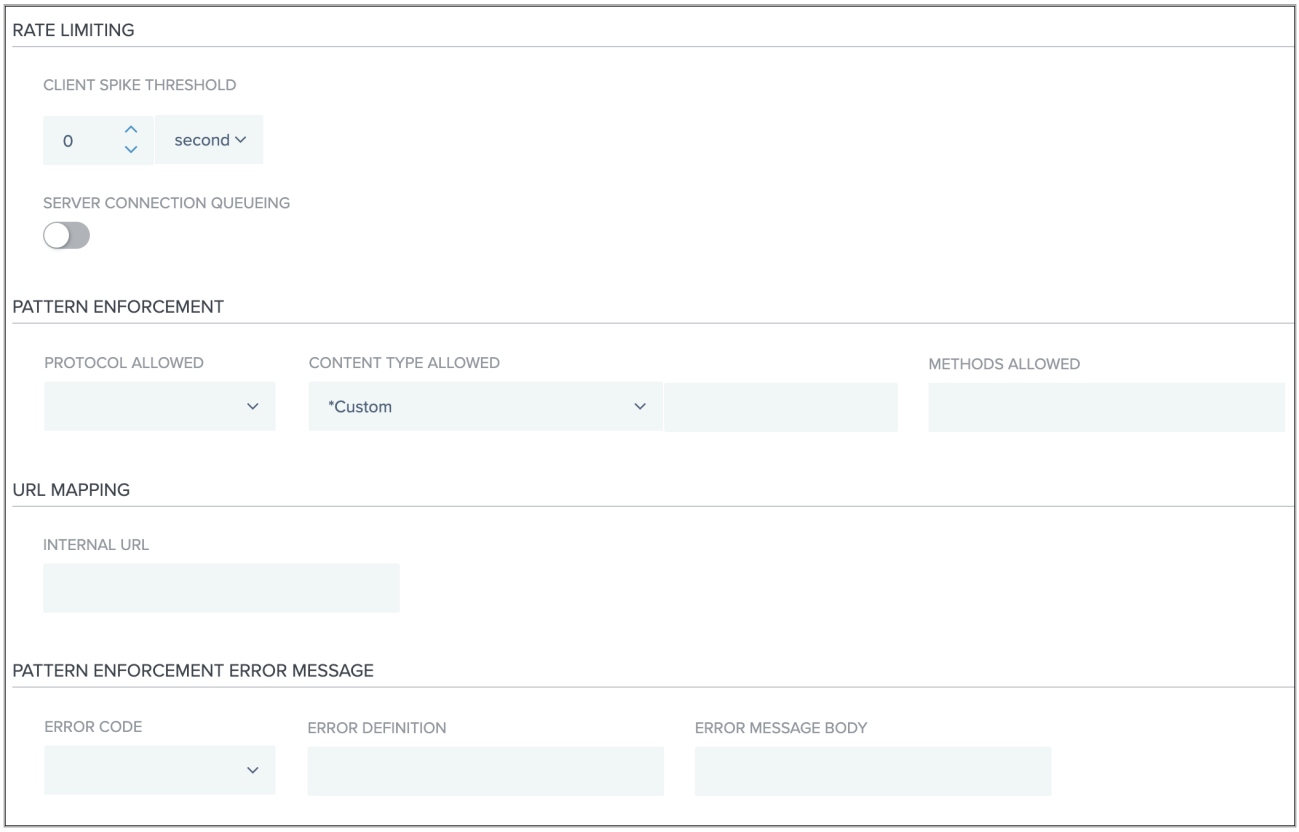
> **Note**
>
> The **Profile** tab also provides option to extract the username from either a JSON Web Token (JWT) token or a custom header. On the dashboard you can select either**JWT** or **Username Header** to configure API JSON, but not both. For more information, see Defining an API using API JSON configuration file in sideband mode.

- Servers

Copyright © 2025 Ping Identity Corporation



- **Inline Security**

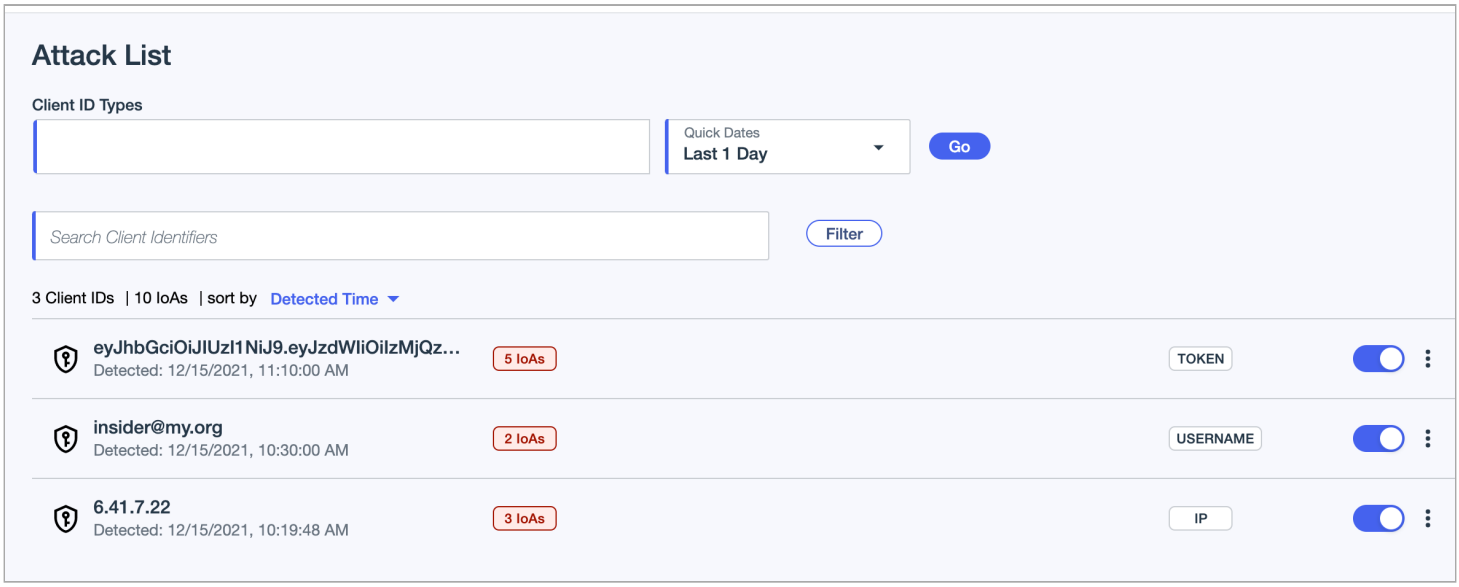


Copyright © 2025 Ping Identity Corporation

Attack management

The Attack management dashboard shows the clients which were flagged for an Indicator of Attack (IoA) for the specified period.

To view the Attack list summary information, click Attack management.



The Attack list has the following columns:

Column	Description
Client ID	The unique ID of the client that originated the IoA.
IoAs	The number of IoAs for the client for the time range.
Client type	The type of client: • Token • IP address • Cookie • Username • API key
Reviewed	Reviewed status toggle: • Reviewed (On) • Not reviewed (Off)

Copyright © 2025 Ping Identity Corporation

Column	Description
Actions	Possible actions to take (three-dots) drop down: • Client activity • Tune IoA detection • Remove from blocklist

Sorting and filtering

Sort the Attack list output according to one of the following:

- Detected time (default), from the most recent date and time to the least recent.
- IoA count, ordered by Client ID, from the client with the highest number of IoAs to the client with the least IoAs.

Apply filters to narrow down the Attack list.

- Select one or more Client ID Types from the drop down menu:
  - Token
  - IP address
  - Cookie
  - Username
  - API key
- Select a date range from the Quick dates drop down menu:
  - Last 1 day (default)
  - Last 7 days
  - Last 30 days
  - Custom: define a period from a starting date and time to an ending date and time

Click Go to apply the filters to the Attack list output.

You can filter the Attack list further:

- Search client identifiers: Enter search strings or partial strings of the Client ID

> **Note**
>
> - The search is case-insensitive.
> - Wildcard searches, for example using an asterisk (`\*`), are not supported.
> - Use of quotation marks is not supported.
> - Be aware of the use of spaces in a search string. A leading or trailing space can filter out results. A single space is not regarded as multiple consecutive spaces.

Copyright © 2025 Ping Identity Corporation

- Click Filter to apply the following filter parameters:
  - Reviewed
    - All (default)
    - Reviewed
    - Not reviewed
  - Select one or more APIs from the drop down
  - Select one or more IoA types from the drop down

Drill downs and actions

Actions

On the right side of the row in the main Attack management list, or at the top right of the IoAs dashboard, click the three-dots drop down to choose an action option:

- Client activity: Navigate to the Client activity dashboard, for further inspection and analysis of the client's activities during the reported period.
- Tune IoA detection: Select this option to update models to not flag this behavior in the future.
- Remove from blocklist: Select this option to update models to remove this entry from the blocklist.

Drill down

Click on a row to navigate to the client's IoAs (Indicators of Attack) dashboard, for further drill downs, inspection and analysis of the client's activities during the reported period.

Client activity

The Client activity dashboard provides breakdown details about a client's application programming interface (API) activity in the reported period.

You can navigate to the Client activity dashboard in one of the following ways:

- Go to Dashboard* → *Blocklist.

  In the blocked Clients table, click the entry in the Client ID column to navigate to its detailed breakdown in the Client activity dashboard.

- Go to Attack management.

  Choose:

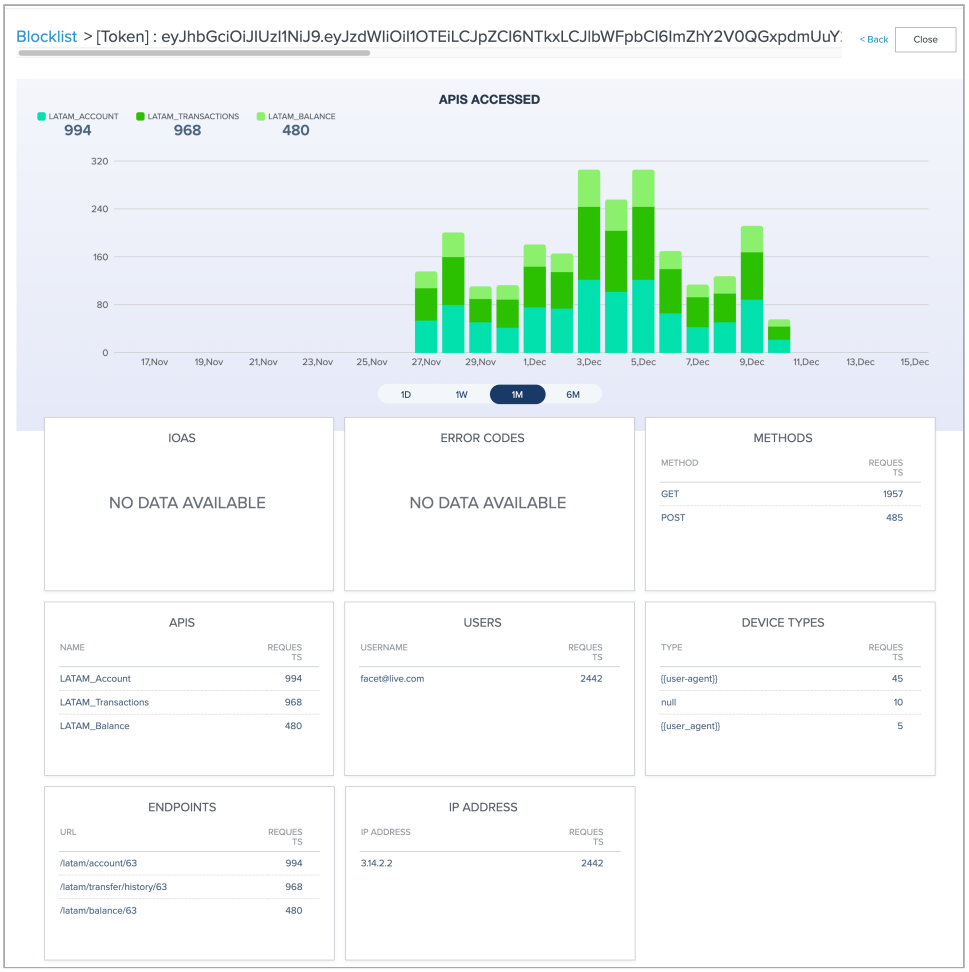
  - On the right side of a client's row in the main Attack management list, click the three-dots drop down, then click Client activity to navigate to its detailed breakdown in the Client activity dashboard.

Copyright © 2025 Ping Identity Corporation

- Click a client's row in the main Attack management list to navigate to the client's IoAs dashboard. At the top of the screen, on the right side of the client's summary line, click the three-dots drop down, then click Client activity to navigate to its detailed breakdown in the Client activity dashboard.

At the top right, Close returns you to the previous dashboard screen that you viewed.

If you navigate to the Client activity dashboard from the Blocklist dashboard, the Blocklist and Back navigation controls also appear at the top of the screen, and will return you to the Blocklist dashboard.



The summary count of APIs accessed by the client for the reported period display in the upper part of the Client activity dashboard. You can adjust the reporting period to display counts of APIs accessed for the past day, week, month or six-month periods.

In the lower part of the screen, the following panes display details about the APIs accessed by the client, for the reported period:

Pane	Description
IoAs	List of the types of IoAs based on the client's activity, and the count of IoAs per type.
Error Codes	List of error codes returned from the client's requests to the APIs.

Copyright © 2025 Ping Identity Corporation

Pane	Description
Methods	List of the API methods used when the client issued API requests, and the count per method.
APIs	List of the APIs receiving requests from the client, and the count of requests per API.
User	List of the usernames issuing requests to the API, and the count of requests per username.
Device Types	List of the device types where requests to the API originated, and the count of requests per device type.
Endpoints	List of the API endpoints that received requests, and the count of requests per API endpoint.
IP Address	List of the Internet Protocol (IP) addresses where requests to the API originated, and the count of requests per IP address.

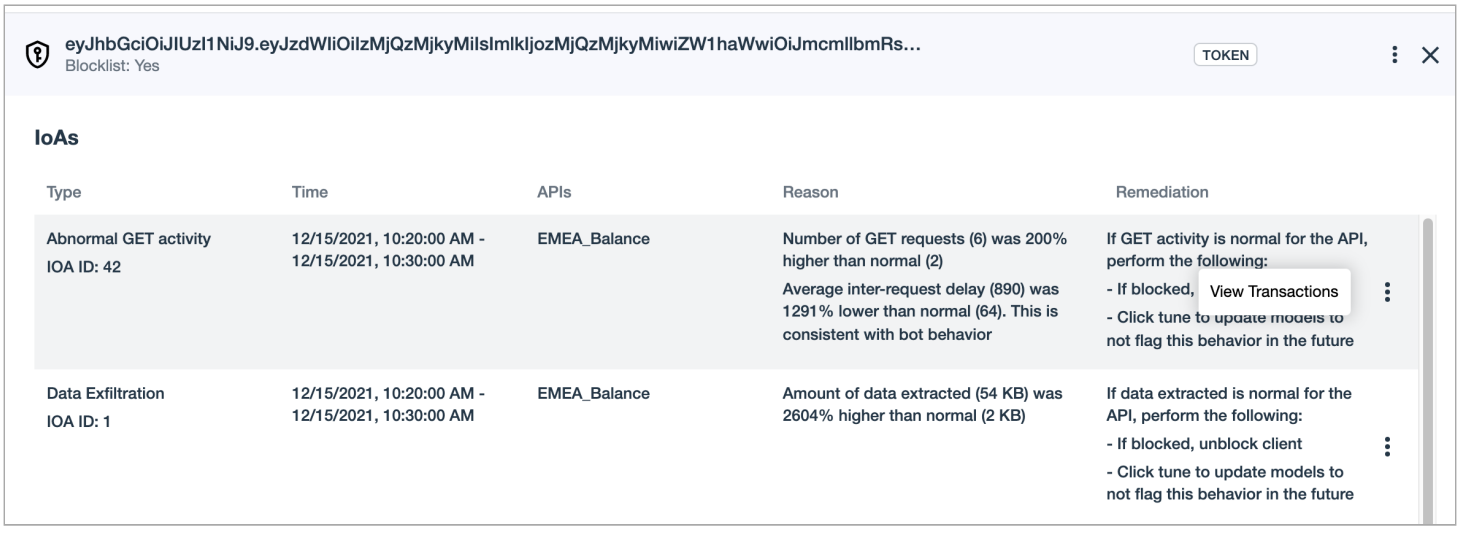
Click Close at the top right, to return to the previous dashboard screen.

IoAs (Indicators of Attack)

The IoAs (Indicators of Attack) dashboard lists the detected IoAs for a client row in the Attack list table of the Attack Management dashboard. The IoAs dashboard provides some high-level details, and functionality for further drill downs, inspection and analysis of the client's activities during the reported period.

Go to Attack management.

Click on a client row in the Attack list table, to navigate to the client's IoAs dashboard, for further drill downs, inspection and analysis of the client's activities during the reported period. The IoAs dashboard lists detected IoAs.



Copyright © 2025 Ping Identity Corporation

Column	Description
Type	Type of IoA.
Time	Starting and ending date and time of the abnormal activity.
API	The name of the impacted API.
Reason	The rationale behind generating the IoA.
Remediation	Suggestions for handling the reported IoA.
Three-dot drop down	Click View transactions to navigate to the Transactions page, for details on each transaction that generated the IoA on the API.

Actions and drill downs

Actions

On the right side of the row in the main Attack management list, or at the top right of the IoAs dashboard, click the three-dots drop down to choose an action option:

- Client activity: Navigate to the Client activity dashboard, for further inspection and analysis of the client's activities during the reported period.
- Tune IoA detection: Select this option to update models to not flag this behavior in the future.
- Remove from blocklist: Select this option to update models to remove this entry from the blocklist.

Drill down

Viewing transactions: To view the list of transactions that generated the IoA, click the three-dot drop down on the right of the IoA row, and then click Viewing transactions. The Viewing transactions dashboard provides functionality for further drill downs, inspection and analysis of the client's activities during the reported period.

Click X in the top right to return to the previous dashboard.

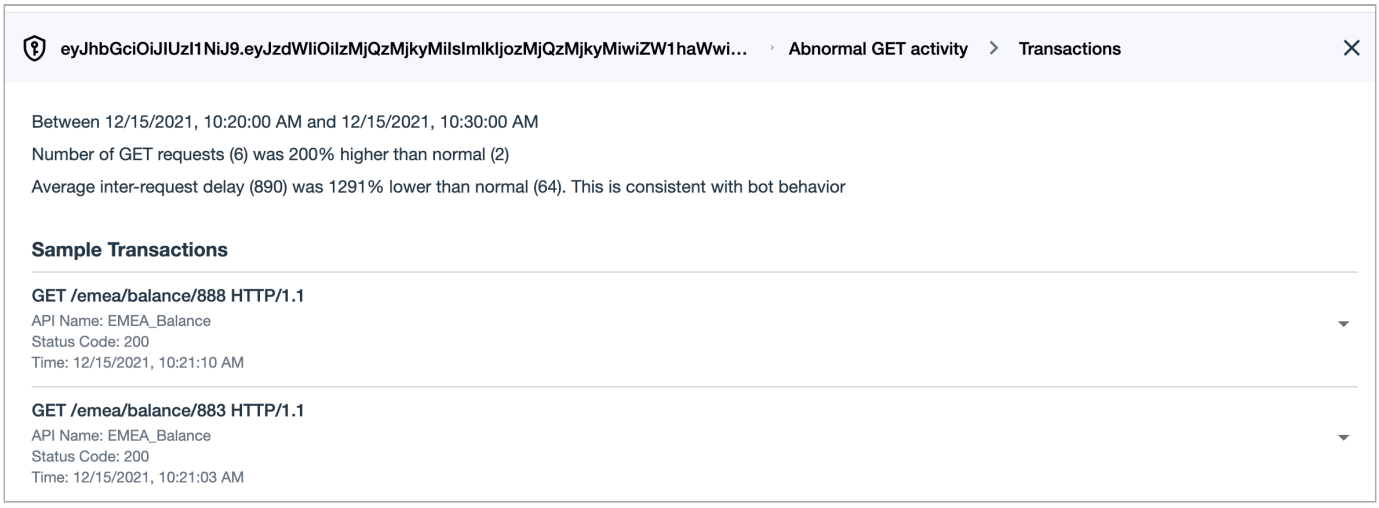
Viewing transactions

Steps

1. Go to Attack management.
2. Click on a client row in the Attack list table, to navigate to the IoAs (Indicators of Attack) dashboard, that lists detected IoAs for the client.
3. To view the list of transactions that generated the IoA on an IoA's row, click the three-dot drop down on the right, and then click View transactions.

Copyright © 2025 Ping Identity Corporation

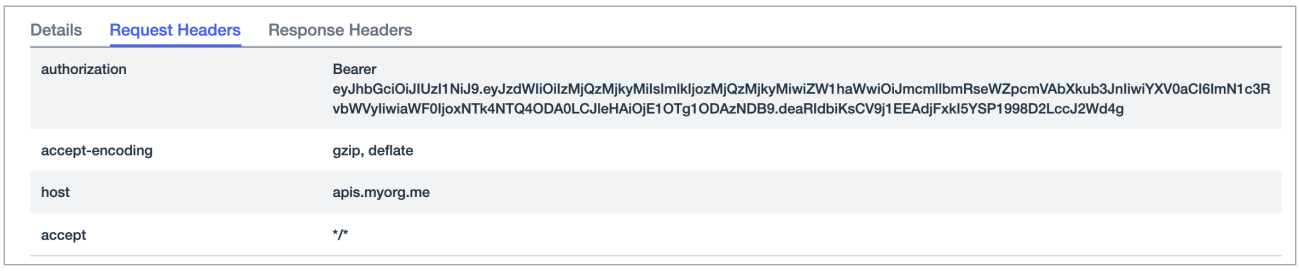
> **Note**
>
> The list of transactions appears with one of the headings:
>
> - `Transactions` : The list displays all the transactions involved in the IOA detection.
> - `Sample Transactions` : When the number of transactions is large, the list displays a sampling of the transactions involved in the IOA detection.



4. Click the drop down arrow on the right of a transaction to view parameter values in the tabs:

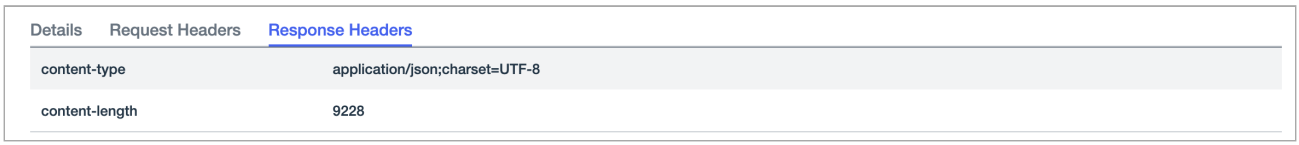
   - Details tab: View parameter values of the transaction's host name, username, token, IP address, response content type and response payload size.

   - Request headers tab: View the transaction's request header parameter values. Examples include authorization type, accept-encoding, host name, and accept response's content type.



Copyright © 2025 Ping Identity Corporation

- Response headers tab: View the transaction's response header parameter values. Example include content type and content length.



5. Click X at the top right, to return to the previous dashboard.

License

The License screen displays PingIntelligence license information for the environment.

PingIntelligence license details include:

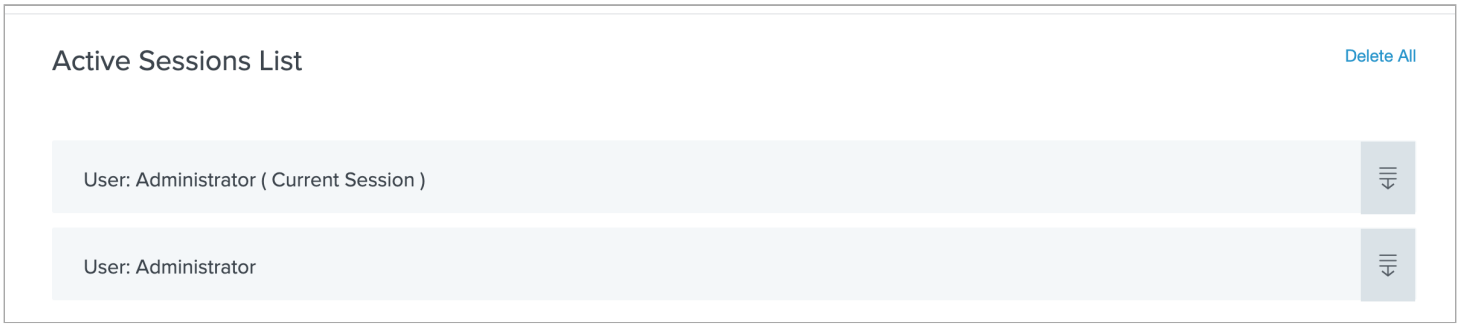
- License type: Subscription or Trial license, and the license expiration date.
- Requests: The number of application programming interface (API) requests processed, in relation to the maximum licensed number of requests per month.

Active sessions

The Active sessions screen displays information about the active users connected to PingIntelligence UI sessions.

The Active sessions screen displays the Active sessions list.

Copyright © 2025 Ping Identity Corporation



In the Active sessions list, your user and session entry is marked `(Current Session)`.

To view a user's session details, expand the user entry in the Active sessions list by clicking the control on the right. The following session parameter values display:

- Role of the user.
- Source IP origin accessing the UI.
- User agent, typically the user's browser and version.
- Created date and time of the user session.
- Last active date and time of the user session.

If there are sessions listed other than the `(Current Session)`, the Delete all control appears, permitting termination of all sessions other than the current session.

Settings

Configure API discovery

To customize the discovery process, configure the discovery parameters on the Dashboard.

Navigate to Settings → Discovered API.

Discovery settings consists of the following three parts:

- Mode - Configure the mode in which application programming interface (API) are published to API Security Enforcer (ASE). The mode can be Manual or Auto.
- Discovery Configuration - Switch discovery on or off, configure the subpath depth of the API base path and discovery interval.
- Default API Properties - Configure the default properties of discovered APIs. You can edit the properties of an individual API in manual mode before publishing it to ASE.

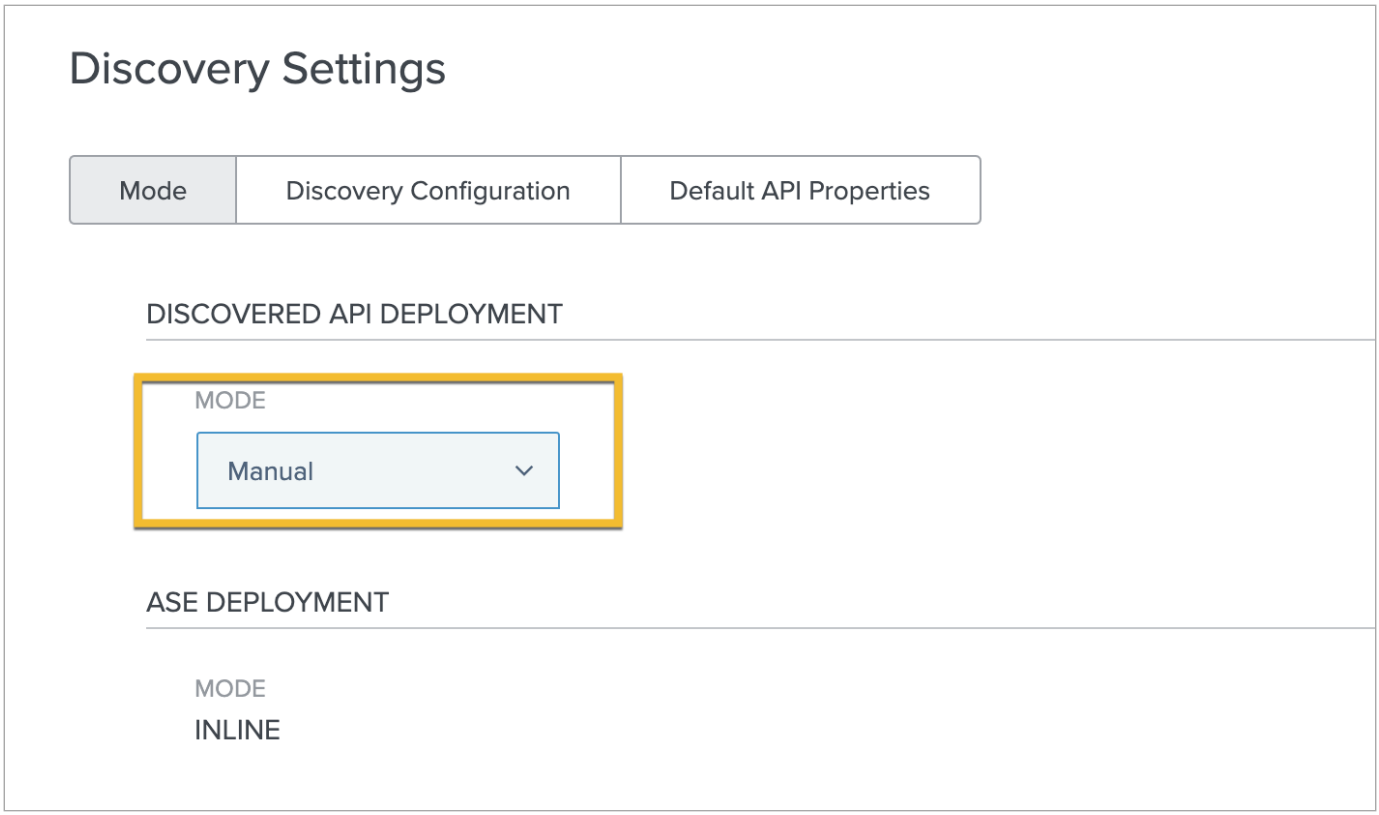
The following sections explain each part of Discovery settings in detail.

Copyright © 2025 Ping Identity Corporation

Mode

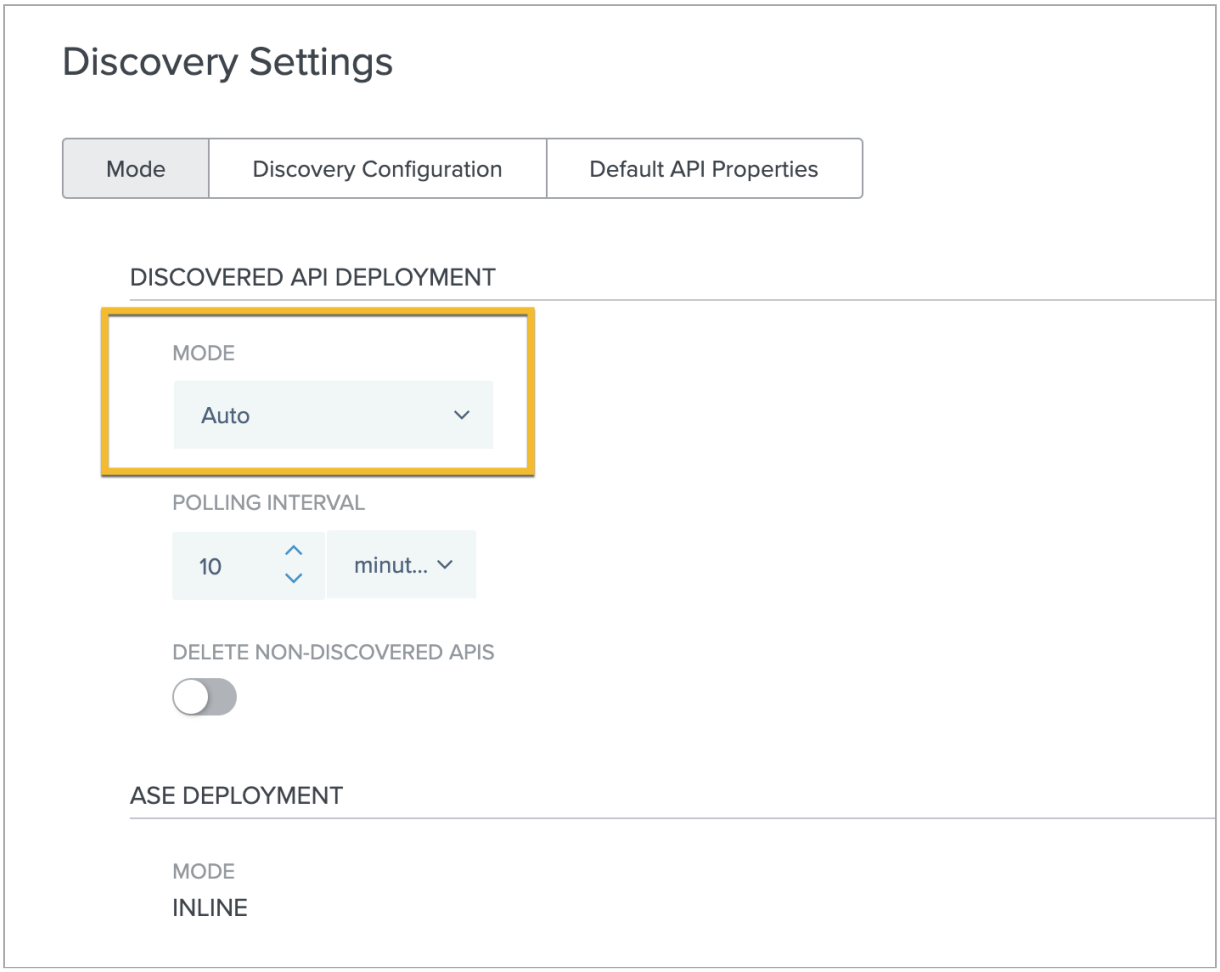
Configure the mode in which dashboard publishes the discovered APIs to ASE. The two modes are:

- Manual: In manual mode, you can review the discovered APIs, edit the properties of the APIs and then publish one or more APIs. For more information on editing the discovered APIs, see Editing discovered APIs.



- Auto: In auto mode, dashboard automatically publishes the APIs after a configured time interval. In auto mode, if you edit an API, it is published in the subsequent interval. Configure the following for auto mode:
  - Polling Interval - The time interval at which dashboard publishes APIs to ASE. It is a good practice to have a minimum of a 10 minute interval.
  - Delete non-discovered APIs - When enabled, any APIs manually added to ASE are deleted.

Copyright © 2025 Ping Identity Corporation



- ASE Deployment - Displays the ASE deployment mode - inline or sideband.

Discovery Configuration

Enable or disable discovery from the Discovery Configuration tab by toggling the AI Engine Discovery button. Configure the following:

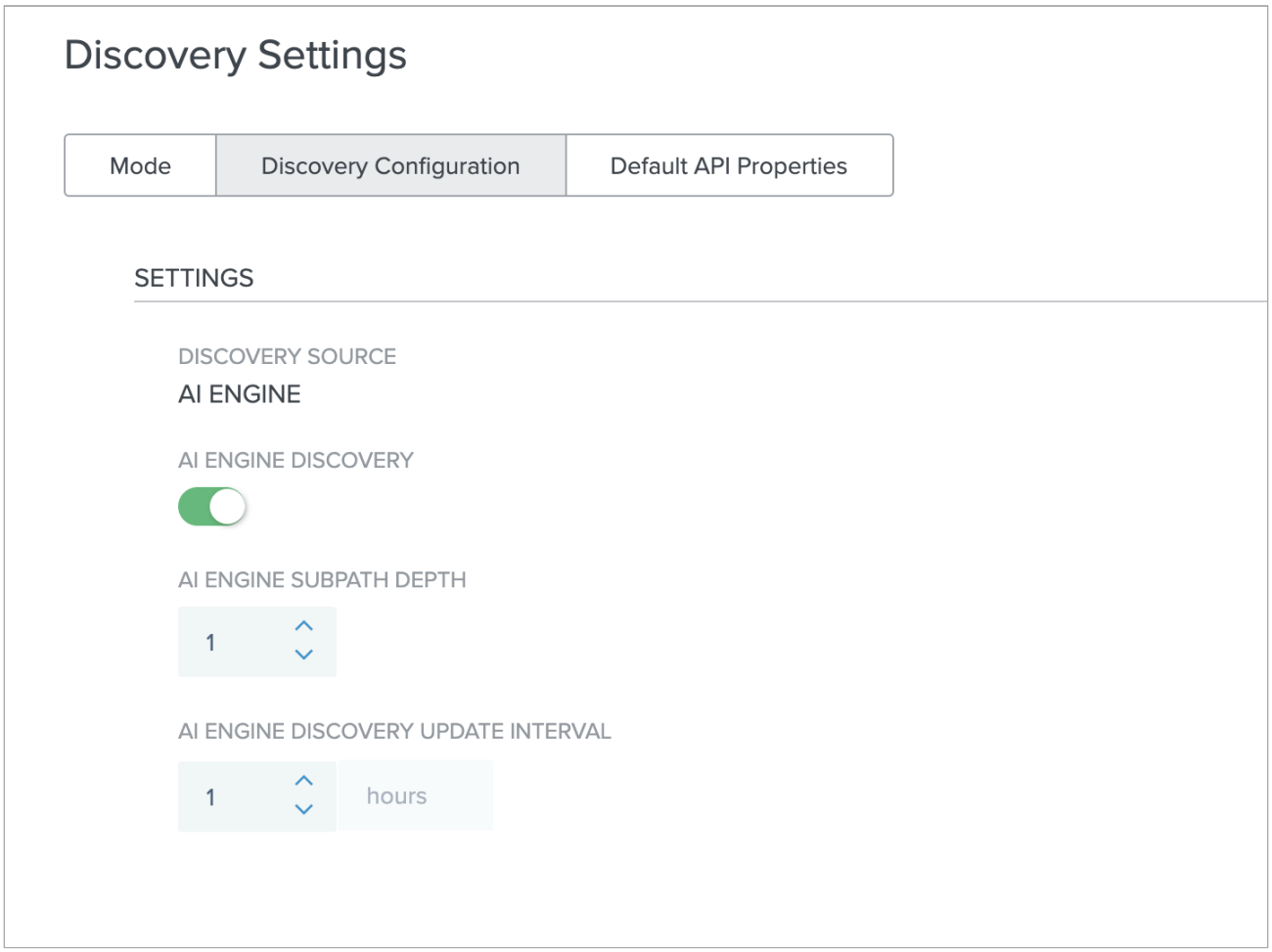
- Discovery Source - the source for newly discovered APIs. Different options are available based on the platform:
  - PingIntelligence for APIs software supports three sources:
    - AI engine
    - PingAccess
    - Axway API gateway

Refer to Configure API discovery for setup instructions.

- AI Engine Discovery - Toggle the button to start or stop API discovery. Make sure a root API is configured in ASE for the AI engine to discover APIs. For more information on discovery process, see API discovery and configuration.
- AI Engine Subpath Depth - Defines the number of subpaths used to uniquely discover the base path of a new API. The maximum allowed value is 6 when ASE is deployed in inline mode and it is 10 when ASE is deployed in sideband mode. For more information, see Discovery sub-paths.

Copyright © 2025 Ping Identity Corporation

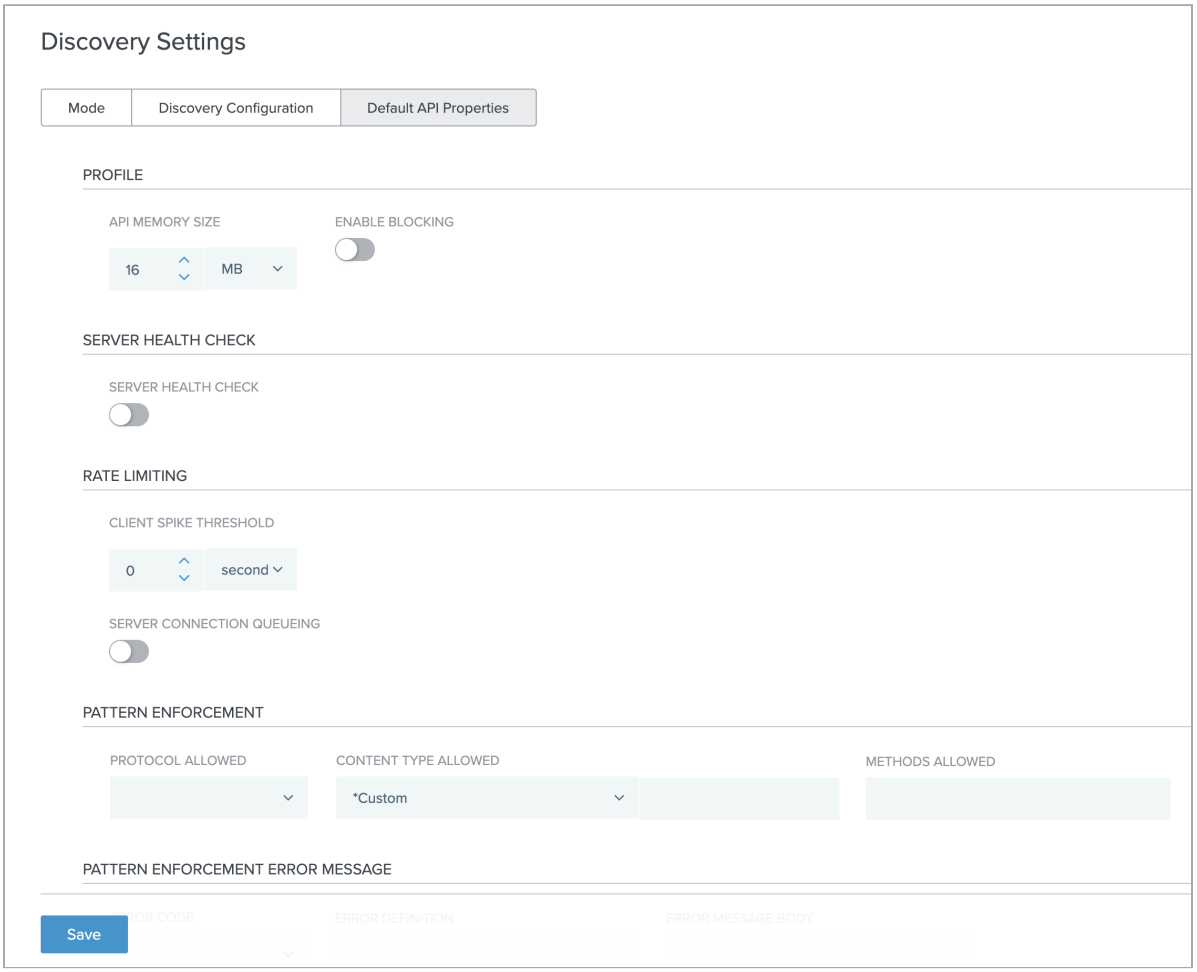
- **AI Engine Discovery Update Interval - Defines the time interval at which new discovered APIs are updated in the Dashboard. The minimum value is 1 hour.**



Default API Properties

You can configure the default API JavaScript Object Notation (JSON) properties from this tab. These properties apply to all discovered APIs. You can edit the properties of the discovered APIs in manual mode before publishing. For more information on the API properties, see Defining an API using API JSON configuration file in inline mode**.**

Copyright © 2025 Ping Identity Corporation



Merging a user-defined API group into the default API group

You can merge a user-defined application programming interface (API) group into the default API group.

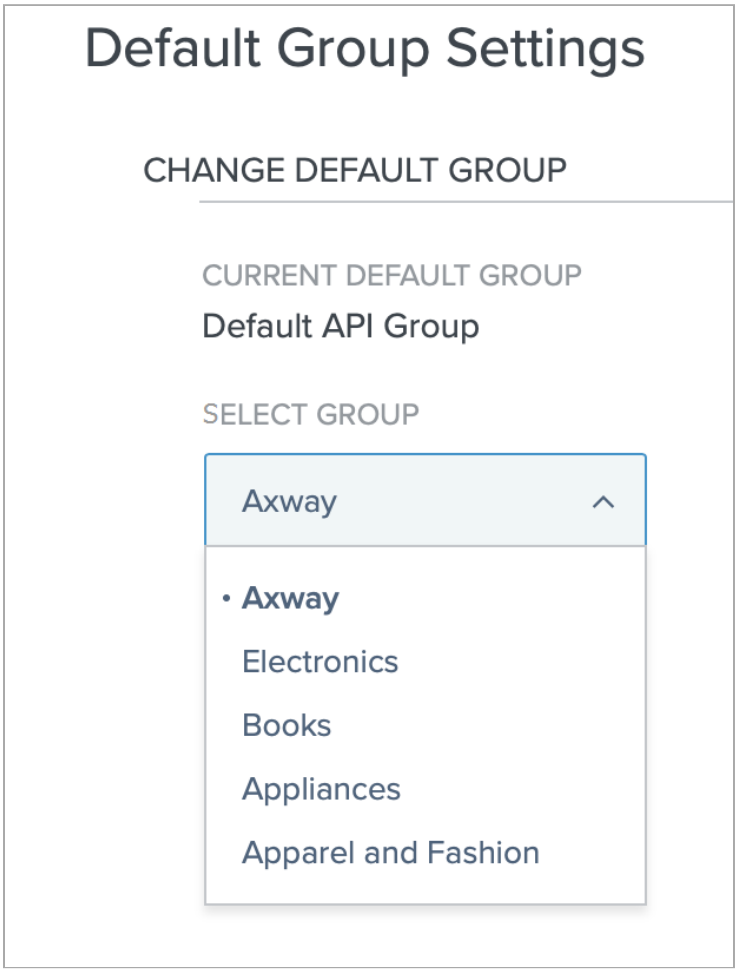
About this task

Complete the following steps:

Steps

1. Click Settings → API grouping settings.
2. From the Select group list, select the API group that you want to move to the default group.
3. Click Save.

Copyright © 2025 Ping Identity Corporation



Configuring training settings

Artificial intelligence (AI) training depends on a set of training parameters in the AI engine. You can configure training variables or reset the trained application programming interface (API) in the AI engine from the Dashboard.

Before you begin

You must have:

- Admin user privileges to configure and update the training settings.

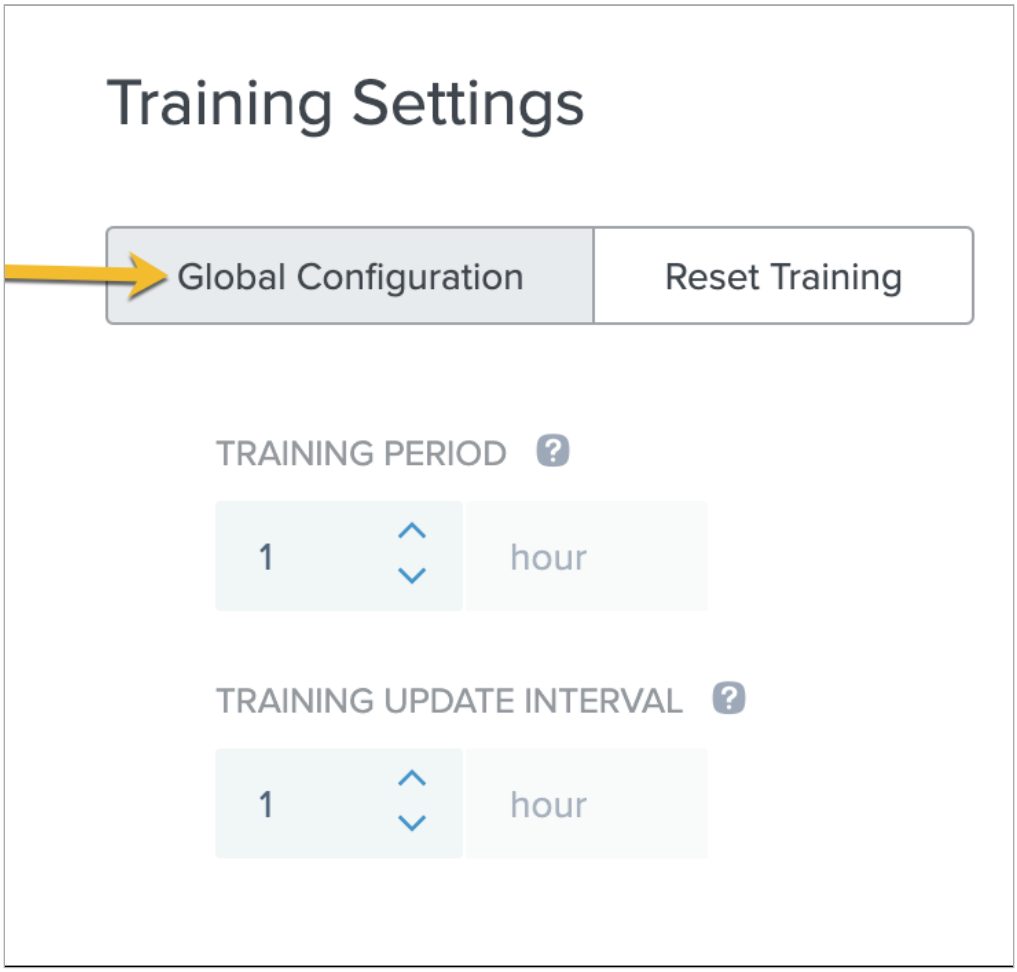
About this task

The following steps provide an overview of the training parameters that can be configured through the Dashboard. It is recommended that you review the variables and configure the best values for your environment.

Steps

- Click Settings → Training Settings. By default, you will reach theGlobal Configuration page.

Copyright © 2025 Ping Identity Corporation



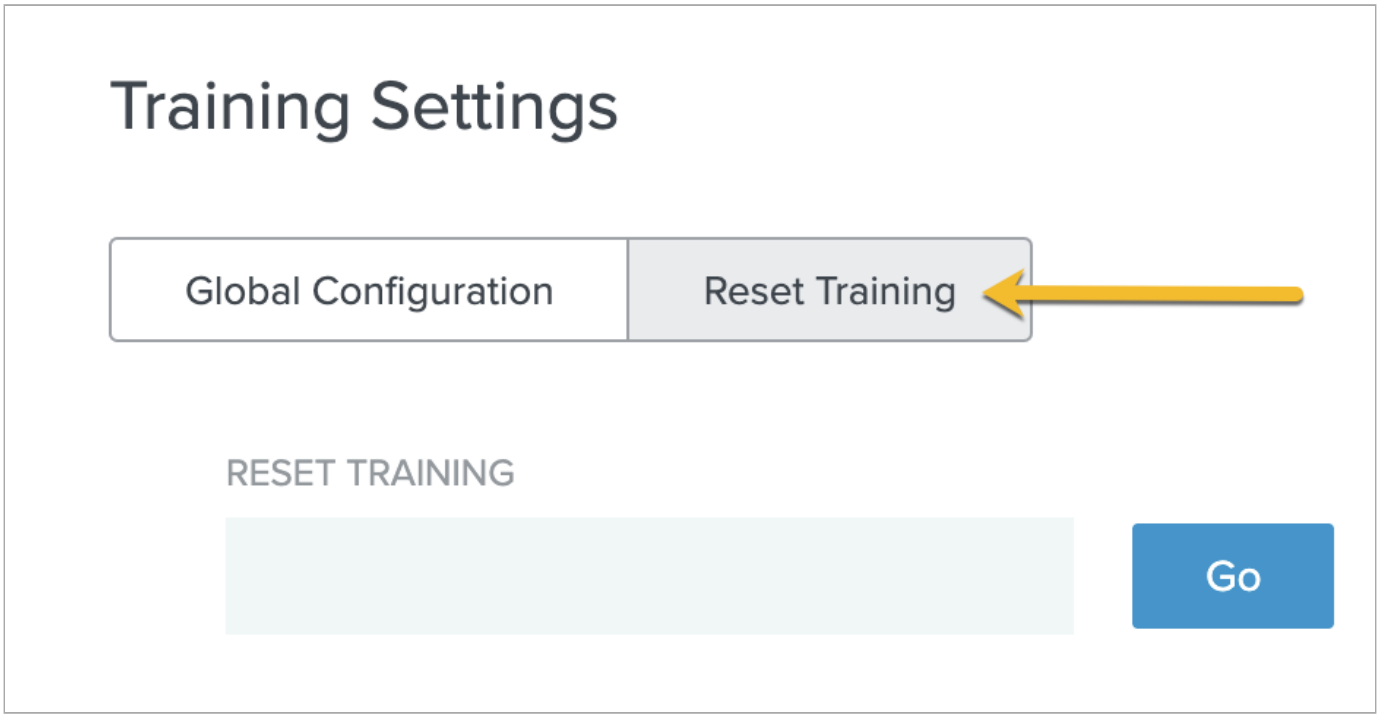
- You can configure the following settings in the Global Configuration page.
  - TRAINING PERIOD: The number of hours to train the AI model before it moves to attack detection mode. It is recommended that you configure the training for at least a week's duration in a production environment.
  - TRAINING UPDATE INTERVAL: The time interval at which continuous learning model thresholds are updated in the AI engine.

These variables are specified in hours with an allowable range of 1 to 10000. Click Save on the bottom-left to reflect the changes.

- You can reset the training of an API or multiple APIs from theReset Training page. To reset the training, select the APIs in RESET TRAINING and click Go.

> **Note**
>
> If there are any pending jobs in the AI engine, the reset will fail with an error notification. You can re-run the reset training after a few minutes. If the reset fails multiple times, follow the steps explained in Resetting trained APIs in ABS to manually reset the API.

Copyright © 2025 Ping Identity Corporation



Related links

- Training the ABS model
- AI engine training variables
- Update the training variables
- Resetting trained APIs

Enabling or disabling attacks

The AI engine detects multiple types of Indicators of Attack (IoAs) on REST application programming interface (API). Each IoA is associated with a unique attack ID. By default all the IoAs are enabled for detection. You can enable or disable detection of a specific IoA, using theEnable/Disable Attacks feature of Attack Management.

Before you begin

You must have:

- **Admin user privileges.**

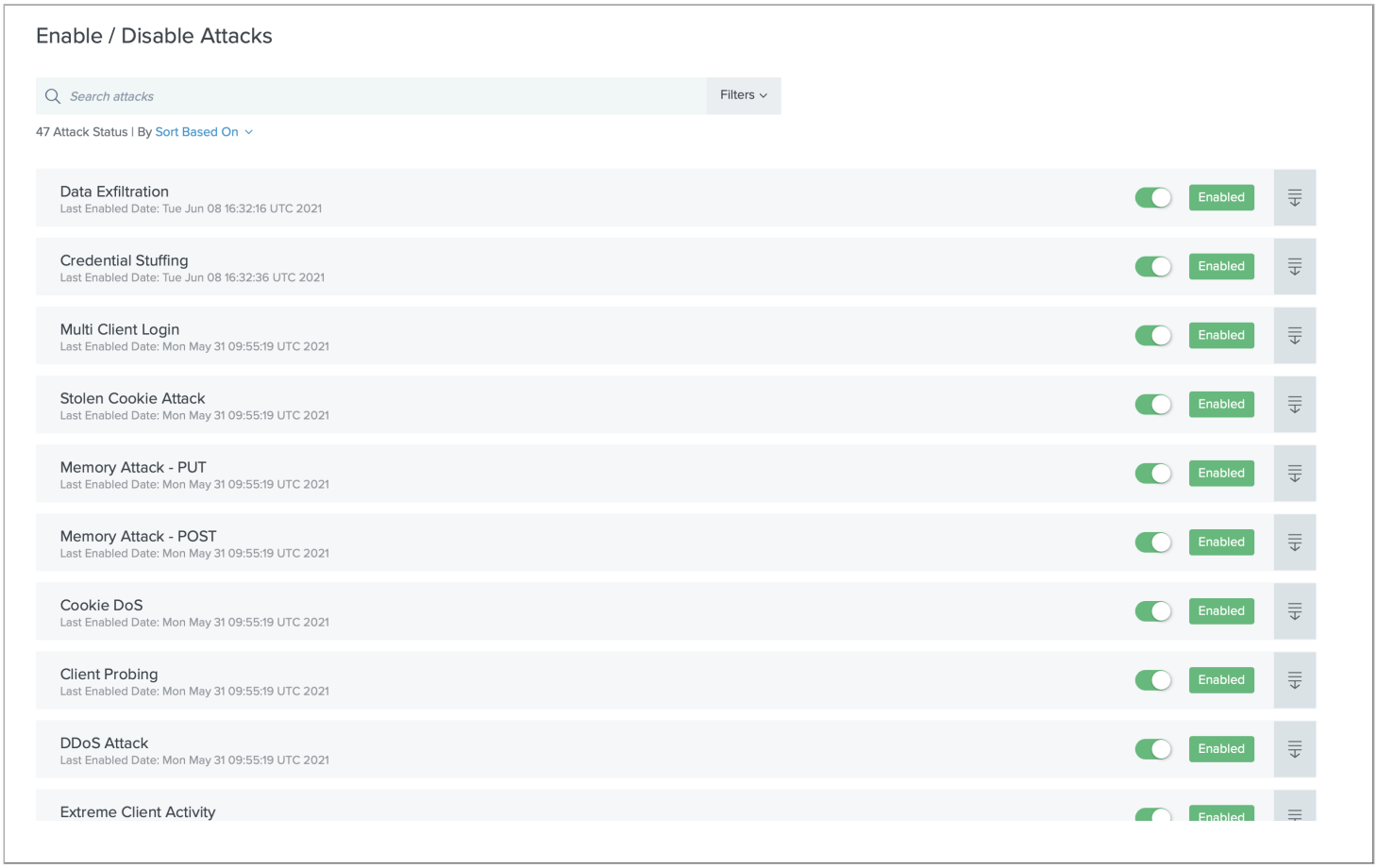
About this task

To enable or disable attacks:

Steps

- **Click Settings → Enable/Disable Attacks.**

Copyright © 2025 Ping Identity Corporation



+

> **Note**
>
> The PingIntelligence Dashboard interacts with the AI Engine when you enable or disable an IoA. If you disable an attack while the AI engine is processing data, it might continue reporting IoAs for a few minutes. The IoA type would be disabled when the next batch of data is processed. When you enable an IoA from the disabled state, the AI engine takes a few minutes to report new IoA events. For more information, see Enable or disable attacks.

- **Click the Toggle**  **to enable or disable an IoA type. The toggle button will not be present if an IoA cannot be disabled. For example, the following IoA IDs cannot be disabled as these are real-time events reported by API Security Enforcer (ASE):**

  - **Attack ID 13: API DDoS Attack Type 2.**
  - **Attack ID 100: Decoy Attack. This IoA ID must be disabled on ASE.**

Copyright © 2025 Ping Identity Corporation

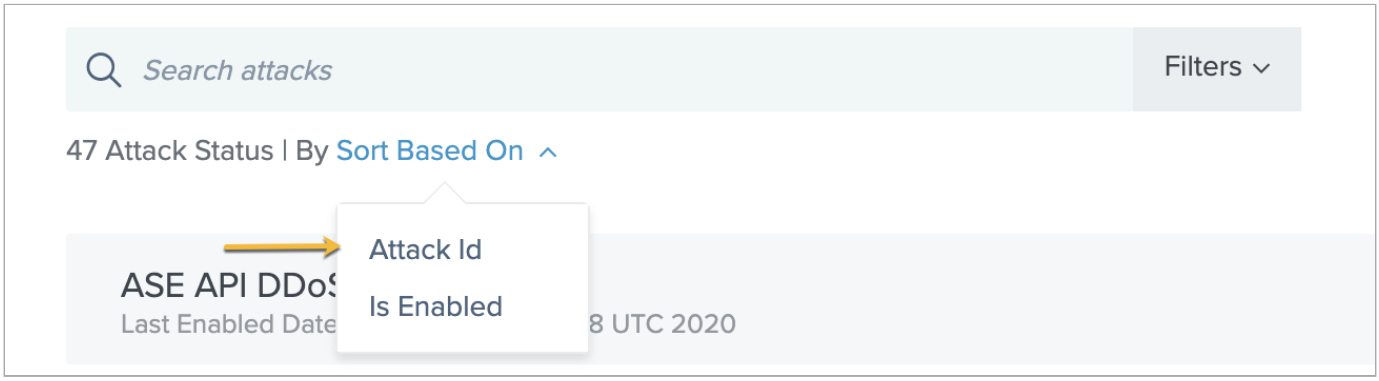
- Attack ID 101: Invalid API Activity. This IoA ID must be disabled on ASE.
  - Click on the Expand  icon for details such as the time the IoA was enabled or disabled. The following screenshot displays the IoA details.

+ image::hwx1640182452025.png[alt="Screen capture of PingIntelligence enable/disable attacks - attack details.",role="border-no-padding"]

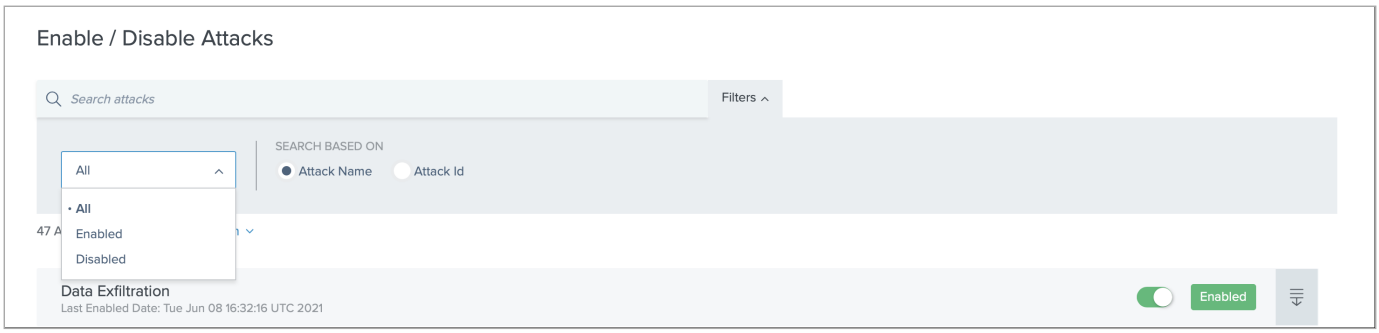
+ You will always be prompted with a confirmation notification before enabling or disabling an IoA. For example when you try to disable an IoA, you will be prompted with the following notification. Click Submit to confirm. You should see a success notification whenever an IoA type is enabled or disabled.

+ image::biq1606234696991.png[alt="Screen capture of PingIntelligence disable attack notification.",role="border-no-padding"]

- Sort the attack types based on IoA ID or Is Enabled status.



- Search based on IoA name or IoA ID within enabled or disabled attacks.



Webhooks

With webhooks, also known as subscriptions, you can use third-party tools to monitor anomalous API activity detected by PingIntelligence.

Use the PingIntelligence Dashboard to create and manage these event subscriptions.

Copyright © 2025 Ping Identity Corporation

Subscriptions are push-based. When an event of interest occurs in PingIntelligence, the event is pushed from PingIntelligence to your monitoring system, such as Splunk and other SIEM systems.

Creating or editing a webhook

You can create a webhook to monitor events in PingIntelligence.

About this task

To create or edit a webhook:

Steps

1. Go to Settings → Webhooks.
2. Click the + icon to add a webhook, or expand an existing webhook and click the Pencil icon to edit it.
3. Enter a descriptive Name for the connection.
4. Enter the destination information.

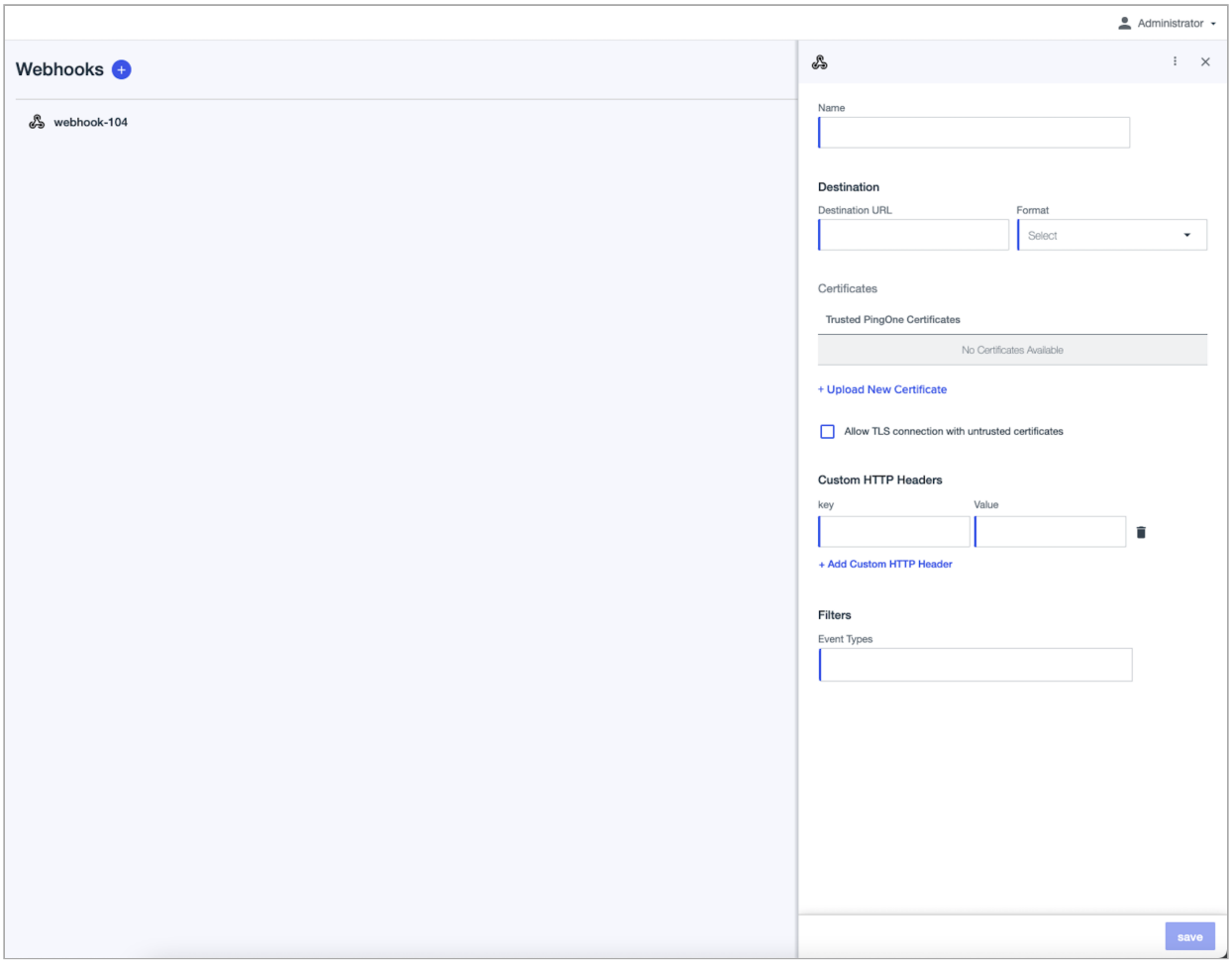
   These settings configure the connection to the monitoring system.

   - Destination URL: The URL of the application that you want to send data to.
   - Format: The format of the activity data. Select the format that is most easily consumed by your management system:
     - Splunk: A Splunk-friendly format.
   - Certificates: A certificate to ensure that the connection is secure. Browse existing certificates, or upload a new one.
   - Allow TLS connection with untrusted certificates: Select this option to allow a certificate that is not from a certificate authority (CA). PingIntelligence certificates, and all certificates signed by the default CAs are trusted. This option is typically used for testing. For more information, see Certificates and key pairs.
   - Custom HTTP headers: Specify additional information for the HTTP headers. You can provide information in the form of key-value pairs.
5. Enter the Filters information.

   These settings determine which events are monitored. Select a category or a subset of events in that category.

   - Event Types: Specify the types of events to monitor: Indicators of Attack (IoAs) created or Anomalies created.
6. Click Save.

Copyright © 2025 Ping Identity Corporation



Enabling or disabling a webhook

If you don't want to delete a webhook, you can disable it in PingIntelligence.

About this task

When you disable a webhook in PingIntelligence, it no longer pushes events to the configured connection. When you re-enable a webhook, it collects events that were missed and pushes them to the monitoring system. Events accumulate for a maximum of two weeks. Events older than two weeks are removed.

Steps

1. Go to Settings → Webhooks.
2. Locate the webhook that you want to enable or disable.
3. In the list, click the toggle to enable or disable the webhook.

Copyright © 2025 Ping Identity Corporation

Result

The toggle moves to the left (gray) when disabled and to the right (blue) when enabled.

Deleting a webhook

You can remove a webhook connection that you no longer need in PingIntelligence.

Steps

1. Go to Settings → Webhooks.
2. Locate the webhook that you want to delete.
3. Click to expand the webhook you want to delete.
4. Click the More Options icon (⋮) to delete the webhook.
5. In the confirmation message, click Delete.

Managing AI engine training

The API Behavioral Security (ABS) AI engine needs to be trained before it can detect attacks on application programming interface (API) services.

The ABS AI Engine training is governed by global variables that are configured using Global configuration update REST API. The AI training runs for the minimum training time set, but a minimum amount of data must also be received before the training period is complete for a given API. You can check the training status by using the Admin REST API.

The ABS AI Engine must be trained on an API before it can be secured. Whenever a new API is added, ABS automatically trains on the new API before looking for attacks.

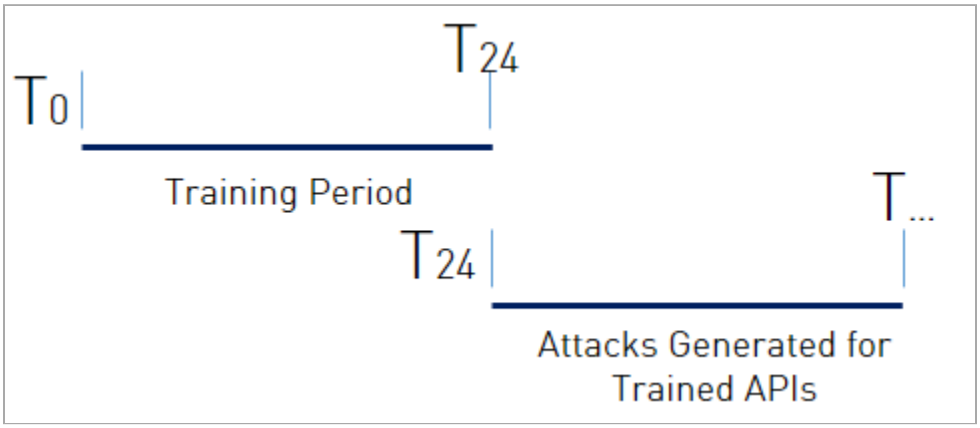
Training the ABS model

The API Behavioral Security (ABS) AI Engine can be trained in a live environment by analyzing API Security Enforcer (ASE) access logs to build its model.

When ABS first receives traffic for a new application programming interface (API), the training period starts. After the defined training period (default is 24 hours) expires, ABS checks if sufficient training data has been collected and will continue training until the models are ready for attack detection. ABS applies continuous learning and adapts its model over time for increased accuracy.

For example, a new API ecosystem is added with four APIs and ABS is configured with a 24 hour training period. Two APIs have immediate API activity, so ABS begins the training period for both APIs. After 24 hours, ABS will detect attacks only for the two trained APIs.

Copyright © 2025 Ping Identity Corporation



If the remaining two APIs start sending traffic three days later, then ABS will begin the 24-hour training period for the remaining APIs and begin attack detection for those APIs at the end of the training period.

> **Important**
>
> It is important to decide on the training and threshold update intervals prior to starting the AI system. Although you can update the training and threshold periods, it is a good practice not to change these variables frequently as this may lead to a change in the behavior of the AI model.

AI engine training variables

PingIntelligence AI training depends on a set of parameters configured using Global configuration update REST API.

These parameters should be configured before starting the system. It is recommended that you review the variables and configure the best values for your environment. Frequent updates to the training variables may lead to a change in behavior of the AI system. The following are the parameters that need to be configured:

- `attack_initial_training`
- `attack_update_interval`
- `continuous_learning`
- `window_length`

The following table describes the various training variables.

Training variables

Variable	Description
`attack_initial_training`	**The number of hours that you want to train the AI model before it moves to the prediction mode. The default value is 24-hours. The minimum value is 1 hour.**

Copyright © 2025 Ping Identity Corporation

Variable	Description
`attack_update_interval`	The time interval in hours at which you would want the model thresholds to be updated. The default value is 24 hours. The minimum value is 1 hour. The value in this variable takes effect only when `continuous_learning` is set to `true`.
`continuous_learning`	Setting this value to `true` configures the AI model to learn continuously based on the live traffic. If it is set to `false`, the AI model detects attack based on the initial training.
`window_length`	The maximum time period that the AI model uses to detect attacks across application programming interface (API)s. The default and maximum value for `window_length` is 24 hours. The training period should be longer than the `windo w_length` period.
`root_api_attack`	Configure as `true` if you want AI engine to detect attacks on the root API. Set it to `false` if you do not wish the AI engine to detect attacks on the root API. The default value is `false`.
`session_inactivity_duration`	The time in minutes for an inactive user session after which API Behavioral Security (ABS) decides that the session has terminated. The default value is 30 minutes. You can configure it to any value in minutes. **Note** This variable only applies to account take over attack.

The following is a snippet from the response global config API:

Copyright © 2025 Ping Identity Corporation

```
{
    “company”: “ping identity”,
    “name”: “api_globalconfig”,
    “description”: “This report contains status information of ABS global configurations”,
    “global_config”: {
        “attack_initial_training”: 2,

        “attack_update_interval”: 1,

        “api_discovery”: true,
        “discovery_initial_period”: 100,
        “discovery_subpath”: 3,
        “discovery_update_interval”: 1,
        “poc”: false,
        “url_limit”: 120,
        “response_size”: 150,
        “continuous_learning”: false,

        “attack_list_count”: 400000,
        “root_api_attack”: true,
        “session_inactivity_duration”: 10

    }
}
```


Miscellaneous variables

Variable	Description
`response_size`	Maximum size in MB of the data fetched by external calls to ABS REST APIs. The default value is 100 MB.
`enable_ssl`	By default it is set to `true`, and SSL communication is enabled between API Security Enforcer (ASE) and ABS, and for external systems making rest API calls to ABS. See Configure SSL on page 10 for more information.

Training period status

API Behavioral Security (ABS) training status is checked using the ABS Admin application programming interface (API) which returns the training duration and prediction mode.

If the prediction variable is set to `true`, ABS has completed training and is discovering attacks. A `false` value means that ABS is still in training mode. The API Uniform Resource Locator (URL) for Admin API is: `https://<ip>:<port>/v4/abs/admin`. Here is a snippet of the Admin API output:


```
"message": "training started at Thu Jul 30 12:32:59 IST 2018",
"training_duration": "2 hours",
"prediction": true
```


Copyright © 2025 Ping Identity Corporation

> **Note**
>
> ABS only detects attacks after the training period is over. During training, no attacks are detected.

Update the training variables

API Behavioral Security (ABS) provides an `update.sh` script in the `/pingidentity/abs/util` directory to update the training related variables.

Using the script, you can update the following variables:

Name	Variable
Continuous learning	`continuous_learning`
Training period	`attack_initial_learning`
Threshold update period	`attack_update_interval`
Window length	`window_length`

You can update the training period when the system is already in a running state by using the script. Review the following use cases before changing the training and threshold period.

> **Caution**
>
> In all the use cases, the default training period is assumed to be 24-hours. If you want to extend the training period, best practice is to add new application programming interface (API)s after the training period is adjusted to avoid APIs completing a shorter training period.

> **Note**
>
> You can also use Global Configuration REST API to update the training variables. For more information, see Global configuration update REST API.

Update the training interval

Increase the training period

You can increase the training period by running the `update.sh` script.

Case	System behavior
The API model is under training, that is, the training period is not over.	In this case, if you increase the training period, for example, from 24 hours to 48 hours, the AI model trains based on the updated training period.

Copyright © 2025 Ping Identity Corporation

Case	System behavior
The API model has completed the training process.	Increasing the training period has no effect on trained APIs. Any new APIs will use the new training period.

Decrease the training period

You can decrease the training period by running the `update.sh` script.

Case	System behavior
The API model is in the training process but has not reached the duration of the new training period.	Decreasing the training period (for example, from 24 hours to 12 hours) shortens the training period to 12 hours for the APIs that have not completed the training process. If the API has completed 10 hours of training, then it will now complete its training period after 2 more hours.
The API model is in the training process and the new training duration is less than the current AI model trained duration.	In this case, the API model stops training itself at the current time and moves to the prediction mode. For example, if the original training period was 24 hours and the AI model has been trained for 18 hours, at this time, if the training period is reduced to 12 hours, the AI model stops training itself and moves to the prediction mode.
API model has completed the training process.	Decreasing the training period has no effect on trained APIs. Any new APIs will use the new training period.

Running the `update.sh` script

About this task

To run the `update.sh` script:

Steps

1. Copy the `update.sh` script from the `util` directory to your MongoDB primary node.

   > **Note**
   >
   > You can find the script in the `/opt/pingidentity/abs/util` directory.

2. Access script help by logging into the MongoDB primary machine and running the following command:

   ```
   /opt/pingidentity/mongo/update.sh help
   ```

   Example:

   For example, change the training period to 48 hours:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/mongo/update.sh -u absuser -p abs123 --attack_initial_training 48
updating training_period to 48
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
The current values of the variables are:
attack_initial_training=48
attack_update_interval=24
api_discovery=false
discovery_update_interval=1
continuous_learning=true
discovery_initial_period=24
url_limit=100
response_size=100
window_length=24
discovery_subpath=3
percentage_diskusage_limit=80

Global Config successfully updated
```

3. After running the script, stop and start all ABS nodes for the updated values to take effect.

> **Note**
>
> You can change the training period and threshold simultaneously or individually.

Tune thresholds for false positives

API Behavioral Security (ABS) automatically generates attack thresholds that are used by the machine learning system to identify attacks and anomalies. Initial attack thresholds are determined based on training and production traffic in your API ecosystem. At the end of the training period, ABS calculates the first set of system-generated threshold values and uses these values to detect attacks.

By default, system-generated threshold values are updated every 24 hours. This frequency can be changed at startup by modifying `attack_update_interval` using Global configuration update REST API or anytime by using the `update.sh` script available in the `util` directory. The minimum value is 1 hour because sufficient traffic is required to update the model.

You can change the threshold period at anytime by running the `update.sh` script. The value of the updated threshold period is applicable immediately. For example, if the current threshold update period is 10 hours and the new threshold period is 12 hours, then the AI model updates the threshold at the 12th hour.

Access script help by signing on to the MongoDB machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```

Example

In the following example, the user has changed the training period and threshold interval together:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/mongo/update.sh -u absuser -p abs123 --attack_initial_training 24 --
attack_update_interval 24
updating attack_initial_training to 24
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
updating attack_update_interval to 24
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
The current values of the variables are:
attack_initial_training=24
attack_update_interval=24
api_discovery=true
discovery_initial_interval=48
```

Checking threshold values

Threshold values can be checked using the ABS Threshold API. For each attack type, one or more variables (for example, Var A, B) are used by the machine learning process during attack detection. All variables have a Normal Threshold Value (tn) and some variables also have an Extreme Threshold Value (tx). These values are used during the attack detection process and automatically update over time to provide improved accuracy.

To view the current threshold settings, use the GET method with the following ABS **`threshold`** API: `https://<ip_address>:<port>/v4/abs/attack/threshold?api=<api_name>`

The IP address and port corresponding to the host ABS machine. The API payload returned is a JSON file which shows the threshold values for each attack type. See Get Threshold API for an example.

Changing attack thresholds

Ping Identity recommends using the automatically generated system thresholds in your production operations. However, if attacks are detected for legitimate traffic (for example, false positives), then manual tuning options are provided. An administrator has two choices:

- Change the system-generated threshold value to a larger user-generated value.
- Disable the variable to stop detecting attacks (see Disabling Attacks)

To identify settings to change, generate an attack report, which includes attacks known to be false positives. For each identified attack, an Attack Code (for example, `varA (Tn), varB (Tn)`) is listed with the threshold variables that triggered the attack. The Attack Code includes the responsible variables (for example, A, B) and threshold types (for example, `Tn`, `Tx`).

The threshold type can be manually adjusted. Ping Identity recommends slowly increasing the triggered threshold values using user-generated thresholds. After each update, evaluate the new setting to see if false positives are reduced. The process can be repeated until the issue is addressed.

The Threshold API PUT method is used to manually override the system generated setting with a user-defined value. When configuring the threshold manually, you can set the normal threshold (tn), the extreme threshold (tx), or either threshold individually.

You can also tune thresholds and unblock clients in the Dashboard to tune threshold values for a specific client identifier. See Attack management.

> **Note**
>
> Make sure that you are in **`Setting thresholds manually`** mode before changing the threshold manually.

Copyright © 2025 Ping Identity Corporation

Changing threshold value Tn only

The `Tn` threshold value can be changed for each attack type for a specific API. The initial `Tx` value is automatically calculated based on the gap between the values of `Tn` and `Tx`. This gap is determined at the end of the training period. The minimum gap is 1 and the value of `Tx` always bigger than `Tn`.

For example, the following values are true at the end of the training period:

- `Tn = 12`
- `Tx = 16`
- `Gap = 4 (Tx-Tn)`

The `threshold` API is used to set `Tn=13` for an API variable. As a result, a gap value of 4 is automatically added to the new `Tn` value, so that `Tx = 17`.

This difference between the value of `Tn` and `Tx` is maintained when only `Tn` is moved. However, the difference between the value of `Tn` and `Tx` can be changed when only `Tx` is changed.

> **Note**
>
> The value of `Tn` can never be more than the value of `Tx`.

Changing Threshold Value Tx only

Change the `Tx` value to adjust the gap between the normal and extreme threshold setting for an attack type on a specific API. The value of `Tx` defines the gap which ranges from a minimum of 1 to the maximum value defined in Threshold range for Tn and Tx. When `Tx` is moved, the system calculated gap calculated at the end of the training period is no longer used. For the attack types where `Tx` is not applicable to the variable, `na` is displayed in the `threshold` API.

> **Note**
>
> If the value of only `Tn` is moved without modifying `Tx`, then the new gap between the value of `Tn` and `Tx` is used until the value of `Tx` is changed again.

Changing threshold value Tn and Tx together

You can change both `Tn` and `Tx` for an attack type on a specific API. When `Tn` and `Tx` are moved simultaneously, the newly defined value of `Tn` and gap for `Tx` are changed. The ranges of `Tn` and `Tx` values are detailed in Threshold range for Tn and Tx.

Configuring threshold value

To manually set a threshold, use the PUT method with the following ABS `attack` API: `https://<ip_address>:<port>/v5/abs/attack/threshold?api=<api_name>`

The IP address and port correspond to the host ABS machine. The API input payload is a JSON file, which sets the threshold value for attack types. The parameters include attack type and Normal Threshold (tn) value. When manually setting the threshold for a variable, ABS `threshold` API displays both system generated and user configured threshold values. ABS applies the user configured threshold values until it is reconfigured to use system generated values (see below).

Copyright © 2025 Ping Identity Corporation

Setting thresholds manually

The `threshold` API with PUT method sets the operation mode for the variable by configuring mode to `system` or `user`.

The following snippet of the `threshold` API with the PUT method shows how to change the threshold mode from system to user and change value of `tn`, `tx`, or both at the same time. If you do not want to change the value for `tn` or `tx` in user mode, leave the field blank by putting `""` in the `threshold` API body. In the following snippet, the values of `tn` and `tx` are both changed:

```
{
 "api_name" : "atmapp",
 "mode": "user",
 "ioc_threshold": [
 {
 "type": "api_memory_attack_type_2",
 "variable": "A",
 "tn": "9",
 "tx": "12"
 },
 {
 "type": "data_exfiltration_attack",
 "variable": "A",
 "tn": "18",
 "tx": ""
 },
 {
 "type": "data_exfiltration_attack",
 "variable": "B",
 "tn": "18",
 "tx": ""
 },
 {
 "type": "api_memory_attack_type_1",
 "variable": "A",
 "tn": "18",
 "tx": ""
 }
 ]
}
{
 "api_name" : "shop",
 "mode": "user",
 "ioc_threshold": [
 {
 "type": "api_memory_attack_type_2",
 "variable": "A",
 "tn": "13"
 },
 {
 "type": "api_memory_attack_type_2",
 "variable": "B",
 "tn": "10"
 }
}
```

The following snippet shows the API response:

Copyright © 2025 Ping Identity Corporation

```
{
 "message": success: "Thresholds set to user mode for given variables.",
 "date": "Mon Jan 08 15:36:05 IST 2018"
}
```

After a threshold value is manually set, ABS uses the updated user threshold values to detect attacks.

When threshold mode is changed back to `system`, the user-configured values are no longer used or displayed in the threshold API output. The following snippet shows changing threshold to system mode from user mode for two variables associated with an API memory attack:

```
{
 "api_name" : "shop",
 "mode": "system",
 "ioc_threshold": [
 {
 "type": "api_memory_attack_type_2",
 "variable": "A",
 },
 {
 "type": "api_memory_attack_type_2",
 "variable": "B",
 }
}
```

The following snippet shows the API response:

```
{
 "message": success: "Thresholds set to system mode for given variables.",
 "date": "Mon Jan 06 15:36:05 IST 2018"
}
```

Resetting trained APIs

Reset trained application programming interface (API)s using `Reset Trained API` REST API of API Behavioral Security (ABS).

Before you begin

- Make sure that API Security Enforcer (ASE) and ABS AI Engine communication is disabled. The communication between ASE and ABS is disabled so that no new access log files are sent to the AI Engine for processing. The training can be reset only when all the access logs available with ABS are processed.
- Wait for all the access logs available with ABS to be processed.

About this task

Copyright © 2025 Ping Identity Corporation

Use the API with DELETE method when you want to retrain the model with more inclusive API traffic or the API JavaScript Object Notation (JSON) definition has changed in ASE. When an AI model training is reset, all the training data, detected attacks for those APIs and the generated thresholds are lost. However, the metrics data is retained even after the API is retrained. Using the `Reset Trained API`, you can retrain one or more than one API at the same time. If ABS is deployed in a cluster setup, you can run the API on any of the ABS cluster nodes.

Complete the following steps to retrain the APIs:

Steps

1. Disable access log upload from ASE to ABS by running the following command on the ASE command-line interface (CLI):

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs
   ```

2. Update the API JSON definition in ASE if there are any changes in API.
3. Run the `reset` API in ABS.

   The following is the URL for the `reset` API.

   ```
   https://<ABS_host>:<ABS_port>/v4/abs/reset
   ```

   - Method: DELETE
   - Body:

     ```
     {
     “apis” :[“shop”,“electronics”]
     }
     ```

     Note

     If you run the `reset` API when the ABS AI engine is processing access logs, you get an error message with 409 status code.
     +

     ```
     {
      “error” : “AI engine is processing access logs; try later. To complete the process,
       make sure to disable access log upload from ASE. For more information, see the ABS admin
     guide.”
     }
     ```

4. Wait for the ABS AI Engine successfully to reset the APIs.

   Result:

   You receive the following success message:

Copyright © 2025 Ping Identity Corporation

```
{
 “status” : “API training reset is successful”
“apis” : [ “shop”, “electronics”]
}
```

Disable attack detection

If you want to disable attack detection for a specific API, tune the user threshold to a maximum value. This follows the same process as changing the attack threshold and sets the user-generated normal threshold value to the maximum for the attack type. Refer to Threshold range for Tn and Tx for a list of maximum values.

When the normal threshold is set to maximum, the machine learning system will not generate attacks based on that variable. All other variables continue to operate in either `system` or `user` mode.

You can also disable or enable an attack ID globally by using the `attackstatus` REST API. For more information, see Enabling or disabling attack IDs.

API discovery process

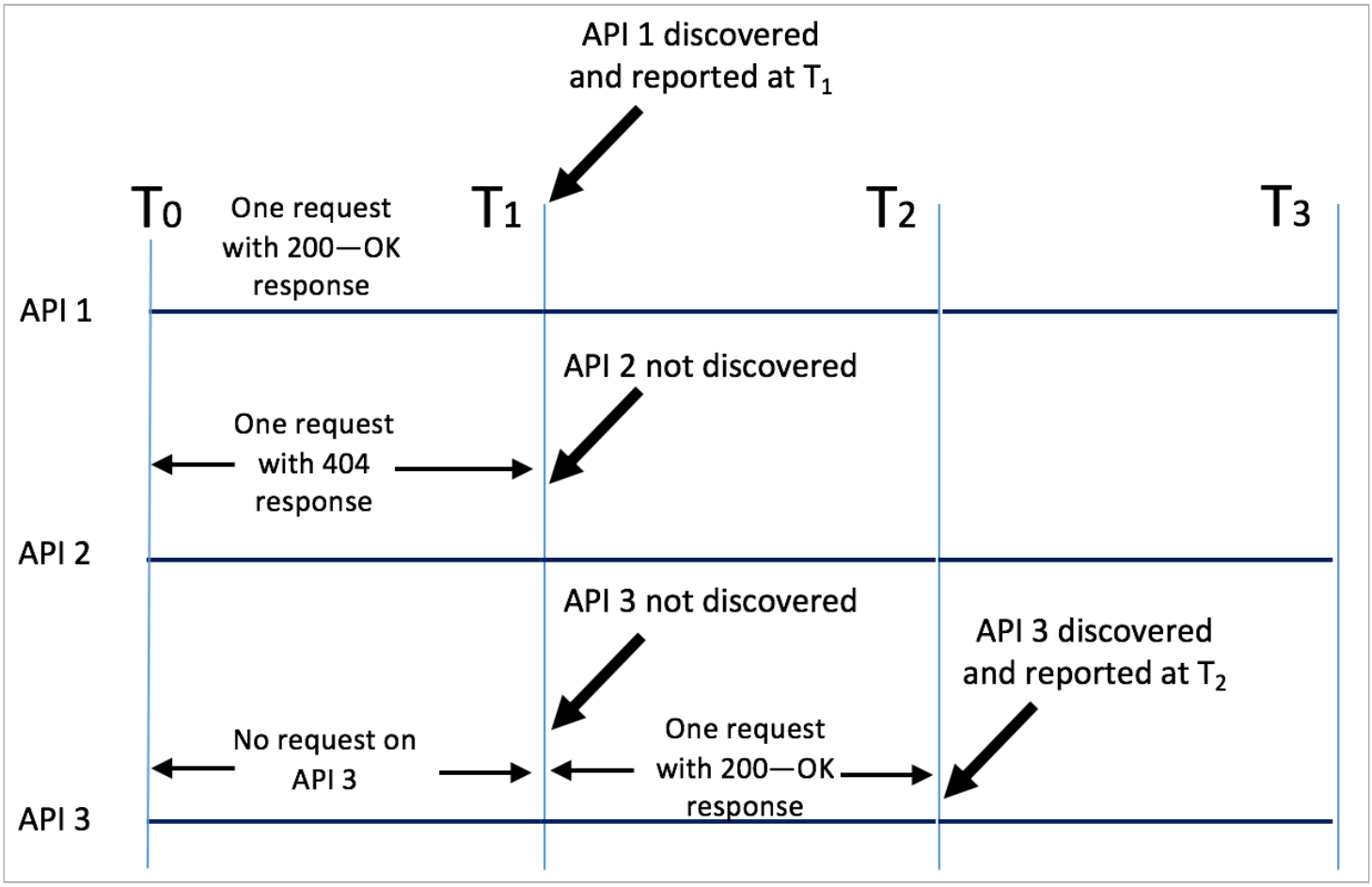
API Behavioral Security (ABS) discovery process starts when the API Security Enforcer (ASE) sends the access log files to ABS.

The discovery process and reporting interval are defined by the variables configured using Global configuration update REST API.

1. ABS processes the ASE log files and looks for new APIs. During the discovery period, ABS monitors the traffic on the API JSON (root API) and requires only one valid request to report an API. ABS considers only valid (200-OK response) requests for discovering APIs. At the end of the discovery period, ABS publishes the discovered APIs. ABS specifically looks for the following four values in the incoming traffic on the root API:
    - Hostname
    - Pathinfo
    - Scheme or protocol
    - Backend server. If ASE is deployed in a sideband mode, then backend server is not reported.
2. At the end of the initial discovery period, ABS does one of the following:
    - If the API definition was learned, then ABS outputs the discovered APIs with the parameters as detailed in the [table_discovery_parameters] below.
    - If the API definition is incomplete, then ABS repeats the discovery process (Step 1) for a `discovery_update_interval` (default is 1 hour).

The following illustration shows an example of the API discovery process:

Copyright © 2025 Ping Identity Corporation



The illustration shows three APIs, API 1, API 2, and API 3 are the undiscovered APIs in your environment. The traffic for these APIs is coming through the root API configured in ASE. The following points explain the discovery process:

- API 1 receives a request in the initial training period with a 200-OK response. This API is discovered at the end of `discovery_initial_period` T1.
- API 2 receives one invalid request (404 response) during the initial discovery period. This API is not reported at T1.
- API 3 did not receive any request in the initial discovery period. Hence it was not reported at T1. However, API 3 got one valid request (200-OK response) in the time-period T1-T2, hence it was reported at time T2. The time period T1-T2 is `discovery_update_interval`.

> **Note**
>
> The initial discovery period applies only to fresh installation of PingIntelligence components. If you are upgrading an existing deployment, the `discovery_update_interval` applies.

ABS API definition reports include the following information for each discovered API:

Information	Description
`host`	Hostname or IP address that is serving the API.

Copyright © 2025 Ping Identity Corporation

Information	Description
`basePath`	The base path on which the API is served. The base path is relative to the host. The value starts with a leading / (slash).
`schemes`	API protocol - value must be HTTP, HTTPS, WS, or WSS.
`consumes`	A list of MIME types that the APIs can consume.
`produces`	A list of MIME types that the APIs can produce.
`paths`	Relative paths to the individual endpoints.
`responses`	Placeholder to hold responses.
`backendHosts`	Backend servers for the API.
`server_ssl`	Value is `true` if backend API server supports encrypted connection. Set to `false` if the backend API server does not support encrypted connection.

You can add the discovered APIs automatically to ASE using Discovered APIs in PingIntelligence for APIs Dashboard. Note that when the root API is configured with the token, cookie, or API key parameter, PingIntelligence will expect all discovered APIs to use the defined identifiers for authentication. If this is not the case, then add the discovered APIs manually in ASE.

API discovery and configuration

The API Behavioral Security (ABS) AI Engine works in tandem with API Security Enforcer (ASE) to automatically discover new and unknown application programming interface (API)s in your ecosystem.

You can view the discovered APIs by using the ABS discovery REST API. You can also add the discovered APIs to ASE by using API Discovery in the PingIntelligence for APIs Dashboard. For more information, see Discovered APIs.

Configuring API discovery

About this task

To configure API discovery in your environment:

Steps

1. Enable ABS in ASE.
2. Define `root` API JavaScript Object Notation (JSON) in ASE.

> **Note**
>
> ABS discovers APIs only for a `root` API JSON in ASE.

3. Optionally, configure OAuth token and API Key parameters in `root` API JSON.
4. Configure discovery related parameters using Global configuration update REST API.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> Use the `update.sh` script to edit the default values related to API discovery. For more information on update script, see Managing discovery intervals.

Configuring ASE for API discovery

About this task

The following table summarizes the variables related to API discovery that you need to configure.

API discovery variables

Variable	Description
`api_discovery`	Set this variable to true to switch on API discovery. To switch off API discovery, set it to false. The default value is true.
`discovery_initial_period`	The initial time in hours during which APIs are discovered in your API ecosystem. The default and minimum value is 1-hour.
`discovery_update_interval`	The time interval in hours at which any new discovered APIs are reported. The default and minimum value is 1-hour.
`discovery_subpath`	The number of sub-paths that is discovered in an API. The minimum value is 1 and maximum value is 6. For more information, see Discovery sub-paths.
`url_limit`	Defines the maximum number of URLs that are reported in a discovered API.

To configure ASE for API discovery:

Steps

- Enable ABS in ASE by running the `enable_abs` command in ASE:

```
./bin/cli.sh -u admin -p admin enable_abs
ABS is now enabled
```

- To verify, run the `status` command in ASE:

Copyright © 2025 Ping Identity Corporation

```
./bin/cli.sh status
API Security Enforcer
status                  : started
mode                    : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : enabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
google pubsub           : disabled
```

- To configure root API in ASE, define `root` API in ASE.

> **Important**
>
> If you have configured other APIs in ASE along with the `root` API, ABS monitors traffic only on the root API for the discovery process. A `root` API in ASE is an API for which the API JSON file has `url` as `"/"` and `hostname` as `"*"`.
> If API discovery is enabled in ABS without `root` API in ASE and you run the `discovery` REST API, it displays an error message: `root API not configured in ASE. To discover APIs configure root API in ASE`.

Example:

The following is a snippet of `root` API JSON:

```
{
    "api_metadata": {
        "protocol": "http",
        "url": "/",
        "hostname": "*",

        "cookie": "",
        "oauth2_access_token": false,
        "apikey_qs": "",
        "apikey_header": "",
        "enable_blocking": false,
        "cookie_idle_timeout": "200m",
        "logout_api_enabled": false,
        "cookie_persistence_enabled": false,
        "login_url": "",
        "api_mapping": {
            "internal_url": ""
        },
```

> **Note**
>
> A sample `root` API ships with ASE in `/pingidentity/ase/config/api` directory.

Copyright © 2025 Ping Identity Corporation

- **Configure API JSON by configuring the settings for `cookie`, `oauth2_access_token`, `apikey_qs`, or `apikey_header` in the `root` API JSON file in ASE.**

> **Note**
>
> API discovery process discovers these parameters in an API only when you set these in the root API. API discovery reports these attributes of an API only when it receives at least 50% of traffic having these attributes. For example, if the root API receives 100 requests and 51 requests have OAuth token, then the OAuth token is reported in the discovered API. Similarly, if the same traffic has less than 50% traffic for API keys or cookies, then they are not reported in the discovered API.

- **Configure API discovery in ABS by setting the `api_discovery` parameter to `true` using Global configuration update REST API.**

> **Note**
>
> If you want update the values on an already running system, use the **`update.sh`** script. For more information on the update script, see Managing discovery intervals.

ABS Discovery API

The Discovery API uses the GET method to display the discovered application programming interface (API) details and is reported only when the `host`, `basepath`, `schemes`, `paths`, and `responses` information is populated.

API Behavioral Security (ABS) provides the following external REST API, which uses the GET method to view the discovered APIs:

URL: `/v4/abs/discovery`

The following is a snippet of the summary output of `discovery` API:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_discovery_summary",
    "description": "This report contains summary of discovered APIs",
    "summary": [
        {
            "api_name": "api_0",
            "host": "bothcookientoken.com",
            "basePath": "/path1",
            "created": "Fri Mar 06 09:29:51:591 2020",
            "updated": "Fri Mar 06 09:50:03:372 2020"
        },
        {
            "api_name": "api_1",
            "host": "path5",
            "basePath": "/path1/path2/path3",
            "created": "Fri Mar 06 10:59:38:975 2020",
            "updated": "Fri Mar 06 11:36:45:596 2020"
        },
        {
            "api_name": "api_14",
            "host": "path5",
            "basePath": "/path1/path2/path3/path4/path5",
            "created": "Fri Mar 06 11:59:14:804 2020",
            "updated": "Fri Mar 06 12:18:24:732 2020"
        },
        {
            "api_name": "api_15",
            "host": "pathx",
            "basePath": "/path1/path2/path3/path4",
            "created": "Fri Mar 06 11:59:16:092 2020",
            "updated": "Fri Mar 06 13:19:25:283 2020"
        },
        {
            "api_name": "api_16",
            "host": "pathx",
            "basePath": "/path1/path2/path3/path4/path5",
            "created": "Fri Mar 06 11:59:16:244 2020",
            "updated": "Fri Mar 06 12:18:26:227 2020"
        },
        {
            "api_name": "api_17",
            "host": "path6",
            "basePath": "/path1/path2/path3/path4/path5/path6",
            "created": "Fri Mar 06 11:59:14:952 2020",
            "updated": "Fri Mar 06 12:18:24:876 2020"
        },
        {
            "api_name": "api_19",
            "host": "path7",
            "basePath": "/path1/path2/path3/path4/path5/path6",
            "created": "Fri Mar 06 11:59:15:096 2020",
            "updated": "Fri Mar 06 12:18:25:028 2020"
        },
        {
            "api_name": "api_9",
            "host": "path2",
            "basePath": "/path1/path2",
            "created": "Fri Mar 06 10:59:00:616 2020",
```


Copyright © 2025 Ping Identity Corporation

```
            "updated": "Fri Mar 06 13:19:23:003 2020"
        }
    ]
}
}
```

Each API name (for example, `api_1`) is auto-generated and starts from `api_0`. This API name can be specified in the `api_name` query parameter to request more details as shown in the next example.

URL: `/v4/abs/discovery?api_name=api_1`

The following is a snippet of a discovered API:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_discovery_details",
    "description": "This report contains details of discovered APIs",
    "info": {
        "title": "api_7"
    },
    "host": "127.0.0.1",
    "basePath": "/shop-books3",
    "cookie": "",
    "oauth2_access_token": false,
    "apikey_qs": "",
    "apikey_header": "",
    "schemes": [
        "HTTP/1.1"
    ],
    "consumes": [],
    "produces": [
        "text/html"
    ],
    "server_ssl": true,
    "backendHosts": [
        "127.0.0.1:4001"
    ],
    "backendServers": [
        "127.0.0.1:4001"
    ],
    "username_header": "",
    "jwt": {
        "username": "username",
        "clientid": "client_id",
        "location": "h:authorization:bearer"
    },
    "paths": {
        "/shop-books3": {
            "GET": {
                "produces": [
                    "text/html"
                ],
                "responses": {
                    "200": {
                        "description": "OK"
                    }
                }
            }
        }
    }
}
```


Note

If the API Security Enforcer (ASE) is deployed in sideband mode, then backend host field in the output shows the Internet Protocol (IP) address as `not available: 0`. The backend server field shows the IP address as `0.0.0.0`. For more information on ASE sideband mode, see the ASE Admin Guide.

Copyright © 2025 Ping Identity Corporation

Managing discovery intervals

You can enable or disable discovery and update the discovery interval by using the `update.sh` script available in the `util` directory. If the training period is set to 1 hour, then discovered application programming interface (API)s are reported 1 hour from the time when API Security Enforcer (ASE) sends access logs. You can update these default values using the `update.sh` script.

About this task

To run the `update.sh` script:

Steps

1. Copy the `update.sh` script from the `/opt/pingidentity/abs/util` directory to your MongoDB primary machine.

   > **Note**
   >
   > You can change the training period and threshold simultaneously or individually.

2. Access the script help by logging in to the MongoDB primary machine and running the following command:

   ```
   /opt/pingidentity/mongo/update.sh help
   ```

   Result:

   ```
   /opt/pingidentity/mongo/update.sh --api_discovery true --discovery_update_interval 48
   updating api_discovery to true
   WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
   updating discovery_update_interval to 48
   The current values of the variables are:
   attack_initial_training=1
   attack_update_interval=24
   api_discovery=false
   discovery_update_interval=1
   continuous_learning=true
   discovery_initial_period=1
   url_limit=100
   response_size=100
   window_length=24
   discovery_subpath=3
   percentage_diskusage_limit=80

   Global Config successfully updated
   ```

Discovery sub-paths

Before starting application programming interface (API) discovery, you must configure the sub-path depth to allow the API Behavioral Security (ABS) AI Engine to accurately detect the API environment. Sub-path depth provides the number of sub-paths for a unique API definition.

Copyright © 2025 Ping Identity Corporation

Here are examples of `discovery_subpath` values:

- "1", example: `/atmapp` is the basepath for `/atmapp/zipcode, /atmapp/update`, etc.
- "2", example: `v1/atmapp` is the basepath for `v1/atmapp/zipcode, v1/atmapp/update`, etc.
- "3", example: `v1/cust1/atmapp` is the basepath for `v1/cust1/atmapp/zipcode`, etc.

The `discovery_subpath` parameter is configured using the Global configuration update REST API and it defines the number of sub-paths in the basepath of the API. The default value is set to 1. The maximum allowed value is six when API Security Enforcer (ASE) is deployed in inline mode and it is 10 when ASE is deployed in sideband mode. The `url_limit` parameter defines the maximum number of Uniform Resource Locator (URL)s reported in a discovered API. The default value is 100.

> **Note**
>
> You can also update the discovery sub-path using PingIntelligence for APIs Dashboard. For more information, see Discovered APIs.

You can update the `url_limit` and `discovery_subpath` by running the `update.sh` script. The `update.sh` script is available in the `/opt/pingidentity/abs/util` directory. Copy the script from the `util` directory to your MongoDB primary machine.

> **Note**
>
> After executing the script, stop and start all ABS nodes for the updated values to take effect.

For example, change the `url_limit` to 50:

```
/opt/pingidentity/mongo/update.sh -u absuser -p abs123 --url_limit 50
updating url_limit to 50
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
The current values of the variables are:
attack_initial_training=48
attack_update_interval=24
api_discovery=false
discovery_update_interval=1
continuous_learning=true
discovery_initial_period=1
url_limit=50
response_size=100
window_length=24
discovery_subpath=3
percentage_diskusage_limit=80

You need to restart all the ABS node for your changes to take effect.
```

Update script help is available by logging into the MongoDB primary machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```


Copyright © 2025 Ping Identity Corporation

PingIntelligence Reference Guide

API Security Enforcer

The API Security Enforcer (ASE) supports multiple deployments modes to provide customers flexibility in deploying PingIntelligence for APIs.

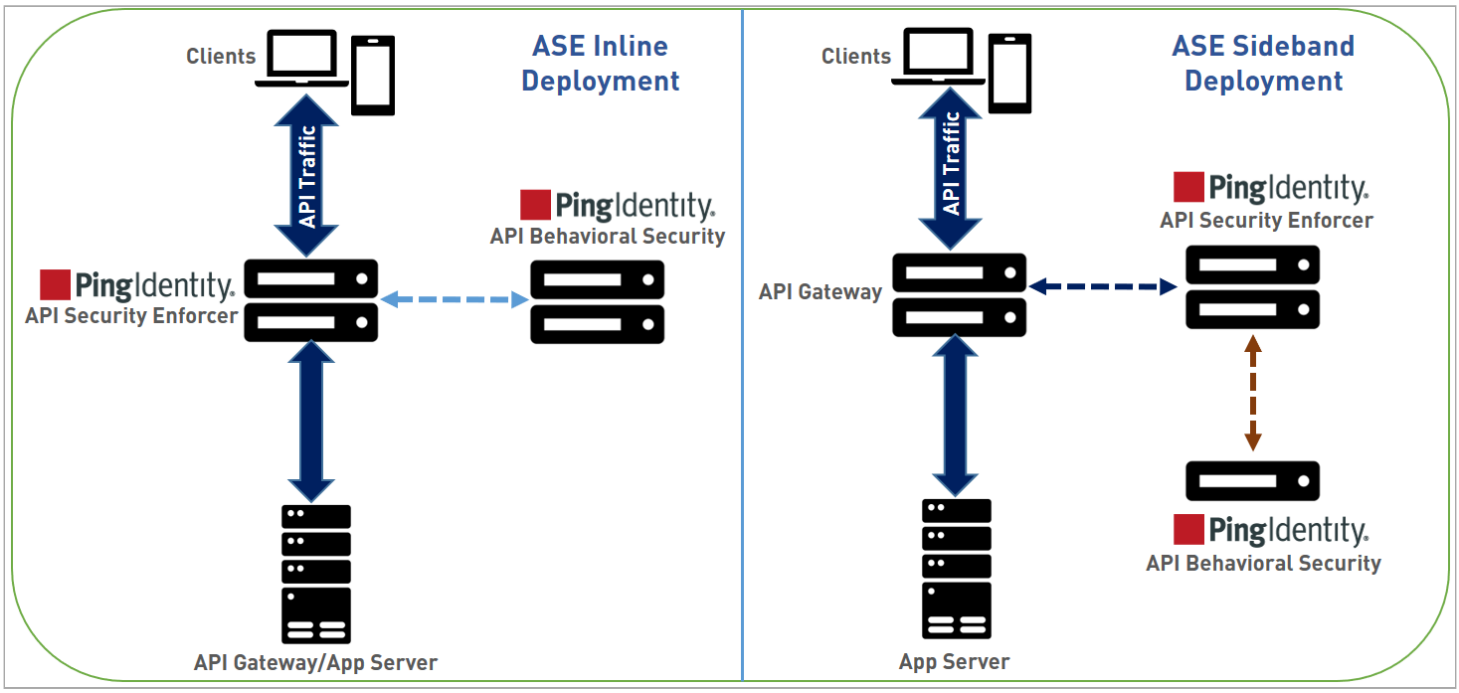
This ASE admin guide covers the following deployment modes:

Inline ASE

ASE receives API client traffic and then routes the traffic to a backend API gateway or directly to App Servers. ASE applies real time security and passes API metadata to the API Behavioral Security (ABS) Engine for AI powered advanced attack detection. ABS engine notifies ASE of attacks, and ASE then blocks the rogue clients.

Sideband ASE

An API gateway receives API client traffic and then makes API calls to pass API metadata to ASE for processing. ASE passes the API metadata to the ABS Engine for AI powered advanced attack detection. ABS engine notifies ASE of attacks, and ASE then works with API gateway to block inbound rogue client requests. See ASE sideband chapter for more information.



The following tables show a summary of security and admin features available in each deployment option.

Copyright © 2025 Ping Identity Corporation

Security Features	Inline	Sideband
Interface to ABS AI Engine for AI powered attack detection.	Yes	Yes
API deception where decoy APIs look like legitimate APIs to hackers. After accessing a decoy API, a hacker is quarantined, plus activity information is collected.	Yes	Yes
Real-time client blocking based on lists with ASE detected attacks, ABS AI Engine detected attacks, or customer-built lists. Blocking can be based on OAuth tokens, API keys, user names, cookies, and IP addresses.	Yes	Yes
Deny and allow list management of tokens, API keys, cookies, IP addresses.	Yes	Yes
Real-time blocking of API clients with traffic that deviates from API attributes.	Yes	No
Dynamic mapping of public API identity to private internal API identity.	Yes	No
Custom API error messages prevent disclosure of sensitive error information.	Yes	No
Admin Features	Inline	Sideband
Simple deployment with modular JSON configuration files.	Yes	Yes
Live updates to add or remove without loss of traffic or stopping services.	Yes	Yes
Obfuscation of keys and passwords.	Yes	Yes
Active-active clustering that supports scaling and resiliency: all nodes are peers and self-learn the configuration, traffic information, and security updates.	Yes	Yes
Syslog information messages sent to Syslog servers in RFC 5424 format.	Yes	Yes
Automatic API discovery discovers API JSON configuration data.	Yes	Yes

Copyright © 2025 Ping Identity Corporation

Security Features	Inline	Sideband
Command-line interface (CLI) and REST API for management and automation tool integration.	Yes	Yes
Linux PAM-based administrator authentication with existing Linux tools.	Yes	Yes
Audit log captures administrative actions for compliance reporting.	Yes	Yes
Distributed inbound flow control limits client traffic and server traffic.	Yes	No
Multiprotocol Layer 7 routing and load balancing of WebSocket, REST API.	Yes	No
Secure connection between ASE and ABS. Secure connection also between ASE and ASE REST APIs.	Yes	Yes

ASE administration

The API Security Enforcer (ASE) is deployed by modifying configuration files to support your environment.

ASE has the following configuration files.

File	Description
`ase.conf`	The master configuration file with parameters to govern ASE functionality.
`cluster.conf`	Configures ASE cluster setup.
abs.conf	Configures ASE to API Behavioral Security (ABS) AI Engine connectivity. ASE sends log files to ABS for processing and receives back client identifiers to block, such as token, IP address, and cookies.

Configuring and updating an ASE license

To start the API Security Enforcer (ASE), you need a valid license.

There are two types of ASE licenses:

Trial license

The trial license is valid for 30 days. At the end of the trial period, ASE stops accepting traffic and shuts down.

Copyright © 2025 Ping Identity Corporation

Subscription license

The subscription license is based on the subscription period.

> **Important**
>
> You should configure your email before configuring the ASE license. ASE sends an email notification to the configured email ID in case the license has expired.

For more information, contact the PingIntelligence for APIs sales team.

> **Note**
>
> If the subscription license has expired, ASE continues to run until a restart.

Copyright © 2025 Ping Identity Corporation

Configuring

Configuring ASE license
About this task

Configure the ASE license:

Steps

1. Request for a license file from the PingIntelligence for APIs sales team.

> **Note**
> The name of the license file must be `PingIntelligence.lic` .

2. Copy the license file to the `/opt/pingidentity/ase/config` directory.
3. Start ASE.

Updating

Updating an existing ASE license
Before you begin

If your existing license has expired, obtain a new license from the PingIntelligence for APIs sales team.

About this task

To update an existing ASE license:

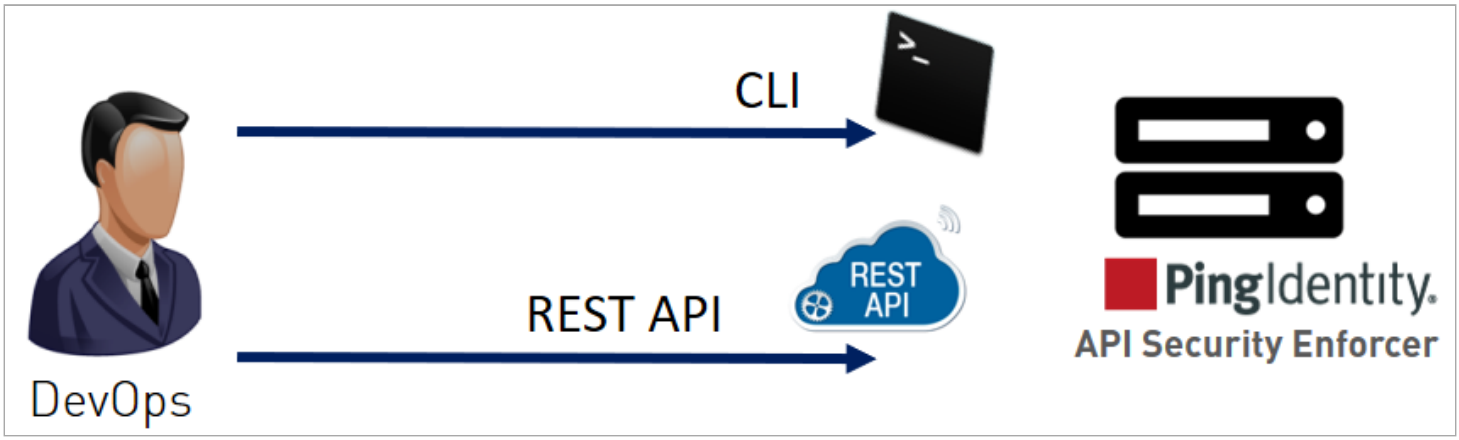
Steps

1. Replace the license file in the `/opt/pingidentity/ase/config` directory.
2. After you have updated the license file, stop and start ASE.

ASE interfaces

There are two interfaces to configure and operate the API Security Enforcer (ASE). You can choose between the command-line interface (CLI) and the ASE REST application programming interface (API).

Copyright © 2025 Ping Identity Corporation



ASE CLI

The `bin/cli.sh` script administers ASE and performs all ASE functions except starting and stopping ASE. To execute commands, enter `cli.sh` followed by the command name. To see a list of all commands, enter the following command into the CLI:

```
/opt/pingidentity/ase/bin/cli.sh
```

The following table lists some basic CLI commands.

Command	Description
`help`	Displays `cli.sh` help. To get a list of CLI commands, run the following: `/opt/pingidentity/ase/bin/cli.sh help`

`version`	Displays ASE's version number. To get ASE's version, run the following:
	`/opt/pingidentity/ase/bin/cli.sh version`
	The system returns the following information:
	`Ping Identity Inc., ASE 3.1.1`
	`Kernel Version : 3.10`
	`Operating System : Red Hat Enterprise Linux Server release 7.0 (Maipo)`
	`Build Date : Fri Aug 24 13:43:22 UTC 2018`

Copyright © 2025 Ping Identity Corporation

Command	Description

`status`	Displays ASE's status. To get ASE's status, run the following:
	`/opt/pingidentity/ase/bin/cli.sh status`
	The system returns the following:
	`Ping Identity Inc., API Security Enforcer`
	`status : started`
	`http/ws : port 80`
	`https/wss : port 443`
	`firewall : enabled`
	`abs : disabled, ssl: enabled`
	`abs attack : disabled`
	`audit : enabled`
	`ase detected attack : disabled`
	`attack list memory : configured 128.00 MB, used 25.60 MB, free 102.40 MB`

`update_password`	Updates the password for ASE admin account.

For a complete list of commands, see CLI for inline ASE and CLI for sideband ASE.

> **Note**
>
> After initial start-up, all configuration changes must be made using `cli.sh` or ASE REST APIs. This includes adding a server, deleting a server, adding a new API, and so on.
> After manually editing an operational JSON file, see Defining an API using API JSON configuration file in inline mode.

ASE REST API

The ASE REST API is used to administer ASE or integrate ASE with third-party products. Using the ASE REST API, you can configure ASE and display ASE statistics, including the number of back end servers, the number of APIs, and so on.

The following table lists ASE REST API.

API command type	List of API commands
API	Create API (POST), Read API (GET), List API (GET), Update API (PUT), Delete API (DELETE)
Server	Create Server (POST), Read Server (GET), Delete Server (DELETE)
Session	Read Persistent Connections (GET)
Cluster	Read Cluster (GET)
Firewall	Read Firewall Status (GET), Update Firewall Status (POST)

Copyright © 2025 Ping Identity Corporation

API command type	List of API commands
Flow Control	Read Flow Control (GET), Update Flow Control for API (POST), Update Flow Control of a Server for an API (POST)

ASE ports

The API Security Enforcer (ASE) uses default ports as defined in the table below. If any port configured in the `ase.conf` file is unavailable, ASE will not start.

Port Number	Description
80	Data port in the Security Group for ASE. HTTP and WebSocket (ws) connections. Accessible from any client. **Note** This is not a secure connection. If you are installing ASE as a non-root user, then use a port greater than 1024.
443	Data port in the Security Group for ASE. HTTPS and Secure WebSocket (wss) connections. Accessible from any client. If you are installing ASE as a non-root user, then use a port greater than 1024.
8010	Management port in the Security Group for ASE. Accessible from management systems and administrators. Used by the command-line interface (CLI) and REST API for managing ASE.
8020	Cluster port in the Security Group for ASE. Accessible from peer ASE nodes. Used by ASE internally to set up the cluster.
8080	API Behavioral Security (ABS) port. Used by ASE for outbound connections to ABS for sending access logs and receive attack information.

Warning

The management ports 8010 and 8020 should not be exposed to the internet and are strictly for internal use. Make sure that these ports are behind your firewall.
In an Amazon Web Services (AWS) environment, both management ports should be private in the Security Group for ASE.
If you are setting up the deployment in an AWS environment with security groups, use private Internet Protocol (IP)s for ABS connections to avoid security group issues.

Copyright © 2025 Ping Identity Corporation

Configuring time zone in ASE

You can set up the API Security Enforcer (ASE) in either the `local` or `UTC` time zone by configuring the `timezone` parameter in the `/pingidentity/ase/config/ase.conf` file.

About this task

All of the management, access, and audit logs capture the time based on the time zone configured in `ase.conf` file. If the `timezone` parameter is left empty, ASE runs in the `UTC` time zone by default. The following is a snippet of `ase.conf` for timezone parameter:

```
; Set the timezone to utc or local. The default timezone is utc.
timezone=local

<truncated ase.conf...>
```

To change the time zone in ASE, complete the following steps:

Steps

1. Stop ASE.
2. Update the `timezone` parameter in the `ase.conf` file.
3. Start ASE.
4. Use ASE `status` command to check ASE's current time zone.

   Example:

   ```
   #./bin/cli.sh -u admin -p status
   API Security Enforcer
   status              : started
   mode                : inline
   http/ws             : port 8080
   https/wss           : port 8443
   firewall            : enabled
   abs                 : disabled, ssl: enabled
   abs attack          : disabled
   audit               : enabled
   ase detected attack : disabled
   attack list memory  : configured 128.00 MB, used 25.60 MB, free 102.40 MB
   log level           : warn
   timezone            : local (MST)
   ```

   Important

   If ASE is deployed in a cluster, make sure to configure the same time zone on each cluster node. If you have used automated deployment to deploy PingIntelligence, the automated deployment configures the same time zone on each ASE node.
   If you have used manual installation, then you must manually configure the time zone on each ASE node.

Next steps

Copyright © 2025 Ping Identity Corporation

Make sure that ASE, the API Behavioral Security (ABS) AI Engine, and PingIntelligence for APIs Dashboard are all configured on the same time zone.

Tuning the host system for high performance

The API Security Enforcer (ASE) ships with a script to tune the host Linux operating system for handling high TCP concurrency and optimizing performance.

About this task

To understand the tuning parameters, see the tuning script comments. When running the tuning script, changes are displayed on the console to provide insight into system modifications.

To undo system changes, run the `untune` script.

> **Note**
> You should be a `root` user to run the **`tune`** and **`untune`** scripts.
> If you are installing ASE as a non-root user, run the **`tune`** script for your platform before starting ASE.

The following commands are for tuning RHEL. For tuning Ubuntu, use the Ubuntu tuning scripts.

Steps

- To tune the host system:
    1. In the command line, run the `tune_rhel7.sh` command.

       Example:

       ```
       /opt/pingidentity/ase/bin/tune_rhel7.sh
       ```

       > **Note**
       > If ASE is deployed in a Docker Container, run the **`tune`** script on the host system, not in the container.

    2. Close the current shell after running the `tune` script and proceeding to start ASE.
- To untune the host system and bring the system back to its original state, run the `untune_rhel7.sh` command.

  Example:

  ```
  /opt/pingidentity/ase/bin/untune_rhel7.sh
  ```

Starting and stopping ASE

Before you begin

For the API Security Engine (ASE) to start, the `ase_master.key` must be present in the `/opt/pingidentity/ase/config` directory.

Copyright © 2025 Ping Identity Corporation

If you have moved the master key to a secured location for security reasons, copy it to the `/config` directory before executing the start script.

About this task

> **Note**
>
> You can also run ASE as a non-root user.

Starting ASE

Starting ASE
Steps

1. To check that the `nofile` limit in `/etc/security/limits.conf` is set to at least 65535 or higher on the host machine, run the following command on the ASE host machine:

   ```
   ulimit -n
   ```

2. Change the working directory to `bin` and run the `start.sh` script.

   Example:

   ```
   /opt/pingidentity/ase/bin/start.sh
   Starting API Security Enforcer 4.0.2...
   please see /opt/pingidentity/ase/logs/controller.log for more details
   ```

Stopping ASE

Stopping ASE
Steps

- Change working directory to `bin` and run the `stop.sh` script.

  Example:

  ```
  /opt/pingidentity/ase/bin/stop.sh -u admin –p admin
  checking API Security Enforcer status…sending stop request to ASE. please wait…
  API Security Enforcer stopped
  ```

Changing default settings

For security reasons, you should change the default master key and passwords in API Security Enforcer (ASE).

Copyright © 2025 Ping Identity Corporation

About this task

To change the default values:

Steps

1. Stop ASE by running the `stop.sh` command.

   Example:

   ```
   /opt/pingidentity/ase/bin/stop.sh -u admin –p admin
   checking API Security Enforcer status…sending stop request to ASE. please wait…
   API Security Enforcer stopped
   ```

   Note

   If you try to generate your `ase_master.key` in step 2 without stopping ASE, you see the following:
   +

   ```
   /opt/pingidentity/ase/bin/cli.sh admin generate_obfkey -u admin -p admin
   API Security Enforcer is running. Please stop ASE before generating new obfuscation master
   key
   ```

2. To change the default `ase_master.key`, run the `generate_obfkey` command.

   You must have the `ase_master.key` to obfuscate keys and passwords in ASE.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin generate_obfkey
   Please take a backup of config/ase_master.key, config/ase.conf,
   config/abs.conf, config/cluster.conf before proceeding
   Warning: Once you create a new obfuscation master key, you should
   obfuscate all config keys also using cli.sh obfuscate_keys
   Warning: Obfuscation master key file /opt/pingidentity/ase/config/ase_master.key already exist.
   This command will delete it create a new key in the same file
   Do you want to proceed [y/n]:
   ```

3. To obfuscate all keys and passwords with the new `ase_master.key`, enter the keys and passwords in `ase.conf`, `abs.conf`, and `cluster.conf` in clear text and run the obfuscation commands.

   For more information on obfuscation, see Obfuscating keys and passwords.

4. After a generating a new `ase_master.key`, start ASE by running the `start.sh` command.

   Example:

   ```
   /opt/pingidentity/ase/bin/start.sh
   Starting API Security Enforcer 4.1...
   please see /opt/pingidentity/ase/logs/controller.log for more details
   ```

5. To change the keystore password, run the `update_keystore_password` command.

Copyright © 2025 Ping Identity Corporation

ASE must be running to update the keystore password. The default password is `asekeystore`.

Example:

```
/opt/pingidentity/ase/bin/cli.sh update_keystore_password -u admin -p admin
New password >
New password again >
keystore password updated
```

6. To change the default admin password, run the `update_password` command.

Example:

```
/opt/pingidentity/ase/bin/cli.sh update_password -u admin
Old password >
New password >
New password again >
Password updated successfully
```

You can change the password on a single ASE node and propagate the change to other nodes in the ASE cluster. For more information, see Propagate changed password.

> **Note**
>
> You must update any change in the ASE admin password by adding the new password in the PingIntelligence for APIs Dashboard. Add the new password to the `<pi_install_dir>/webgui/config/webgui.properties` file and obfuscate it.

Obfuscating keys and passwords

Using the API Security Enforcer (ASE) command-line interface (CLI), you can obfuscate keys and passwords configured in the `ase.conf`, `cluster.conf`, and `abs.conf` files.

Before you begin

When obfuscating a password, you must stop ASE.

About this task

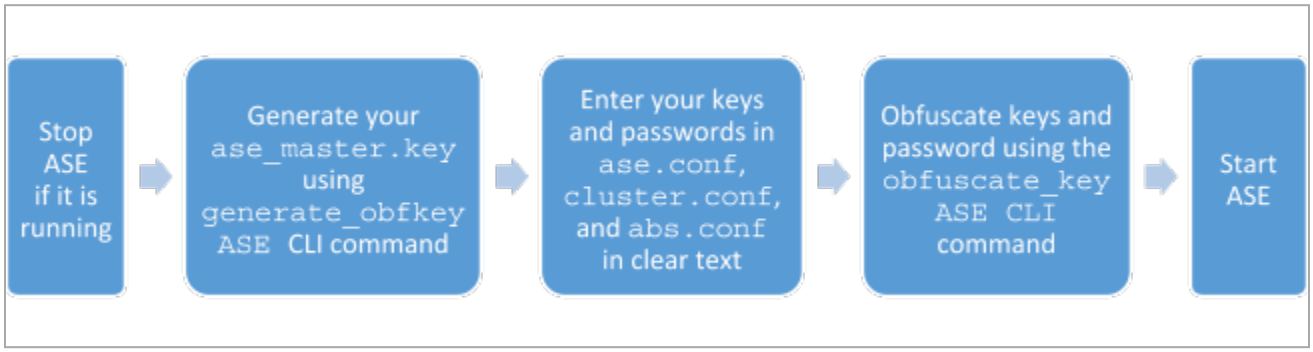
Here is the obfuscated data in each file:

- `ase.conf` – Email and keystore (PKCS#12) password
- `cluster.conf` – Cluster authentication key
- `abs.conf` – ABS access and secret key

ASE ships with a default master key (`ase_master.key`), which is used to obfuscate other keys and passwords. You should generate your own `ase_master.key`.

The following diagram summarizes the obfuscation process.

Copyright © 2025 Ping Identity Corporation



Steps

1. Generate your `ase_master.key` by running the `generate_obfkey` ASE CLI command.

   The new `ase_master.key` is used to obfuscate the keys and passwords in the configuration files.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh generate_obfkey -u admin -p

   Please take a backup of config/ase_master.key, config/ase.conf, config/abs.conf, config/cluster.conf
   before proceeding

   Warning: Once you create a new obfuscation master key, you should obfuscate all config keys also
   using cli.sh obfuscate_keys

   Warning: Obfuscation master key file /opt/pingidentity/ase/config/ase_master.key already exists.
   This command will delete it and create a new key in the same file.

   Do you want to proceed [y/n]:y
   creating new obfuscation master key
   Success: created new obfuscation master key at /opt/pingidentity/ase/config/ase_master.key
   ```

   > **Important**
   >
   > In an ASE cluster, the `ase_master.key` must be manually copied to each cluster node.

2. Obfuscate keys and passwords:

   1. Enter the keys and passwords in clear text in the `ase.conf`, `cluster.conf`, and `abs.conf` files.
   2. Run the `obfuscate_keys` command to obfuscate keys and passwords.

      Example:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh obfuscate_keys -u admin -p

Please take a backup of config/ase_master.key, config/ase.conf, config/abs.conf, and config/
cluster.conf before proceeding

If config keys and passwords are already obfuscated using the current master key, they are not
obfuscated again

Following keys will be obfuscated:
config/ase.conf: sender_password, keystore_password
config/abs.conf: access_key, secret_key
config/cluster.conf: cluster_secret_key

Do you want to proceed [y/n]:y
obfuscating config/ase.conf, success
obfuscating config/abs.conf, success
obfuscating config/cluster.conf, success
```

3. Start ASE after keys and passwords are obfuscated.

> **Important**
>
> After the keys and passwords are obfuscated, the `ase_master.key` must be moved to a secure location from ASE for security reasons. If you want to restart ASE, the `ase_master.key` must be present in the `/opt/pingidentity/ase/config/` directory.

Deleting UUID to propagate a changed password

You can change the password on a single API Security Enforcer (ASE) node and propagate the change to other nodes in the ASE cluster.

About this task

To do this, you must copy the `/data` directory of the ASE node on which the password has been modified to the other nodes in the cluster.

> **Important**
>
> The `/data/ase.store` file stores the password information and the universally unique identifier (UUID) of the ASE node. You must delete the UUID of the ASE node with the modified password before copying its `/data` directory to the other nodes in the cluster. This avoids cluster synchronization issues caused by duplicate UUIDs.

To propagate the changed password to all the nodes in an ASE cluster:

Steps

1. Change the password for the ASE node.

   For more information, see Change Admin password.

2. Stop the ASE node.

Copyright © 2025 Ping Identity Corporation

For more information, see Stop ASE.

3. Run the `delete-uuid` script.

   Example:

   ```
   /opt/pingidentity/ase# ./util/delete-uuid
   Deleting uuid  02cdf7b3-dfb7-4d5b-b9a1-171e89664d11
   Success
   ```

4. Verify the successful deletion of UUID by re-executing the `delete-uuid` script.

   Example:

   ```
   /opt/pingidentity/ase# ./util/delete-uuid
   uuid does not exist in database
   ```

5. Copy the `/data` directory to other nodes in the cluster.

PKCS#12 keystore

The API Security Enforcer (ASE) ships with a default PKCS#12 keystore.

The default password is `asekeystore`. The default password is obfuscated and configured in the `ase.conf` file.

For security reasons, you must update the default PKCS#12 keystore password by using the `update_keystore_password` command.

```
/opt/pingidentity/ase/bin/cli.sh update_keystore_password -u admin -p admin
New password >
New password again >
keystore password updated
```

The password is updated and obfuscated at the same time.

> **Note**
>
> ASE must be running to update the keystore password.

Directory structure

During the installation process, the API Security Enforcer (ASE) creates the following directories.

Copyright © 2025 Ping Identity Corporation

Directory Name	Purpose
`config`	Contains files and directories to configure ASE and its application programming interface (API)s. The `certs` sub-directory contains the keys and certificates for Secure Sockets Layer (SSL)/TLS 1.2.
`data`	For internal use. Do not change anything in this directory.
`logs`	Stores ASE log files including access log files sent to the Application Behavioral Security (ABS) AI Engine for analysis. The access log files are compressed and moved to the `abs_uploaded` directory after they have been uploaded to ABS.
`lib`	For internal use. Do not change anything in this directory.
`bin`	Contains scripts including the start and stop ASE, tuning script for ASE performance. **Note** The scripts in the `bin` directory are not editable.
`util`	The `util` directory contains scripts to check and open ABS ports as well as script to purge logs. • `check_ports.sh` - Check ABS ports • `open_ports_ase.sh:` - Run this script on the ASE machine to open the default ASE ports: 80, 443, 8010, and 8020. • Purge logs

Administering an ASE cluster

An API Security Enforcer (ASE) cluster runs either in a single cloud or across multiple clouds. All ASE cluster nodes communicate over a TCP connection to continuously synchronize the configuration in real time. Cluster nodes are symmetrical which eliminates a single point of failure.

Key features of ASE clustering are:

- ASE node addition to a live cluster without configuring the node – true auto-scaling
- Configuration (`ase.conf` , application programming interface (API) JavaScript Object Notation (JSON) files) synchronization across all cluster nodes
- Update and delete operations using command-line interface (CLI) and REST APIs
- Run time addition or deletion of cluster nodes
- Real-time deny list synchronization across cluster
- A single cluster with nodes spanning across multiple data centers

Copyright © 2025 Ping Identity Corporation

Several cluster features are unique to the deployed environment including:

- Authentication token for API gateway (ASE sideband only)
- Cookie replication across all cluster nodes (ASE inline only)

CLI configuration commands executed at any cluster node are automatically replicated across all cluster nodes. All nodes remain current with respect to configuration modifications. Cluster nodes synchronize SSL certificates across various ASE nodes.

Add or remove a node from the cluster without disrupting any live traffic. The amount of time required to activate a new cluster node is dependent on the time to synchronize the configuration and cookie information from other nodes.

ASE cluster performs real-time synchronization of cookies for ASE inline configurations. This is critical for session mirroring or handling a DNS flip between requests from the same client. Since no master or slave nodes exist, all cluster nodes synchronize cookie information – which means that each node has the same cookies as other nodes.

ASE also synchronizes `ase.conf` files across cluster nodes with the exception of a few parameters: data ports, management ports, and number of processes.

ASE cluster deployment

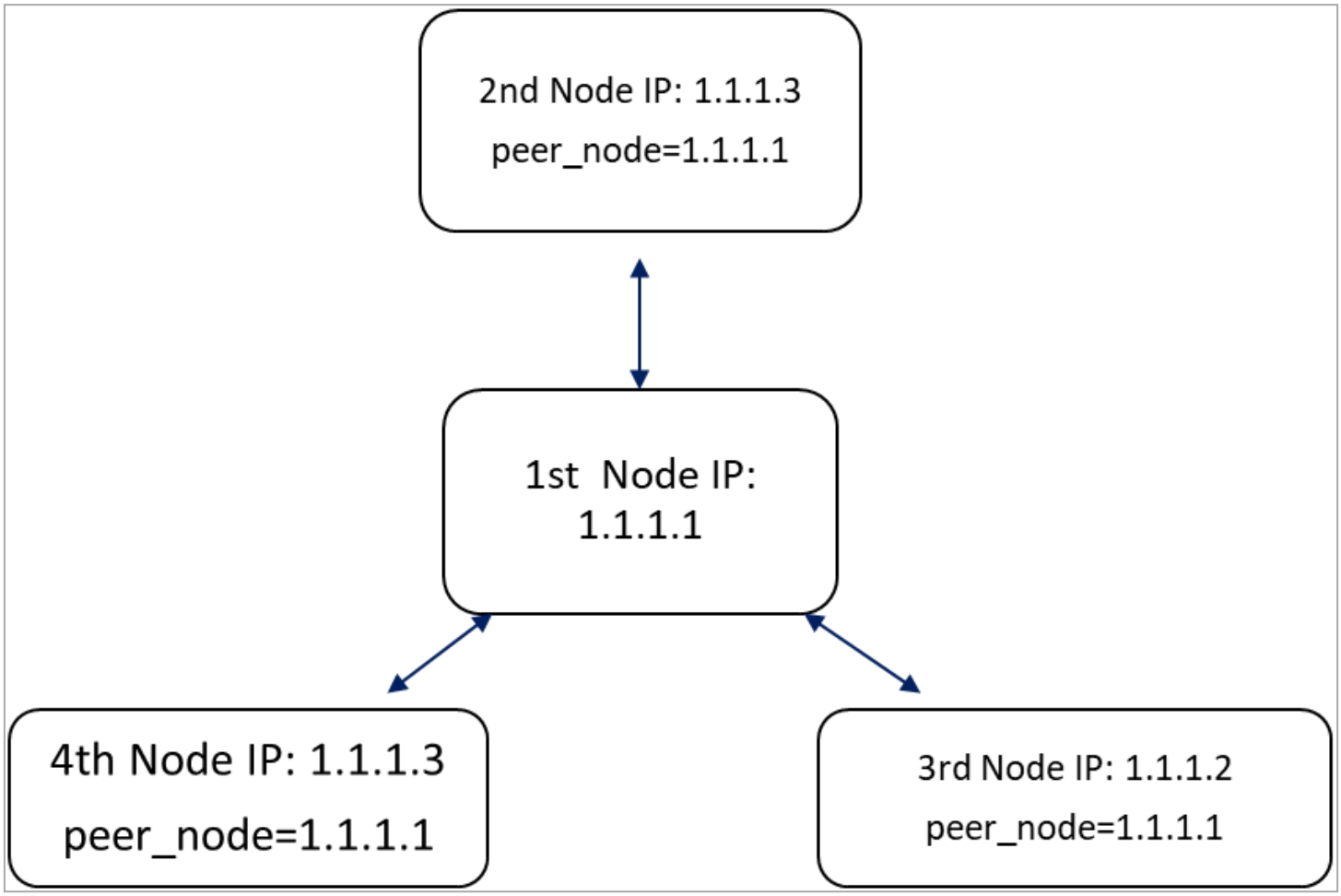
ASE cluster is a distributed node architecture. Ping Identity recommends that one cluster node be designated the management node through which all configuration changes are performed. This helps maintain consistency of operations across nodes. However, no restrictions exist on using other nodes in the cluster to make changes. If two different nodes are used to modify the ASE cluster, then the latest configuration change based on time-stamps is synchronized across the nodes.

ASE cluster uses a circular deployment. During setup, the first node of the cluster acts as the central node of the cluster from which all cluster nodes synchronize configuration and cookie data. When the setup of all nodes is complete, the nodes communicate with each other to synchronize the latest session information.

> **Note**
>
> If the first node or management node goes down, the functioning of the other cluster nodes is not affected. Make sure the peer node provided in the `cluster.conf` file is running before adding a new node.

When an ASE cluster is setup, the `peer_node` parameter must be configured with an IPv4 address and port number. ASE uses this value to connect to other nodes of the cluster. To add new cluster nodes, activate one node at a time. In the following example, the `peer_node` Internet Protocol (IP) address for all nodes is the IP address of the first node. Each node must wait until the process of adding the previous node is completed.

Copyright © 2025 Ping Identity Corporation



Use the status command to verify status before adding the next node in the cluster.

```
/opt/pingidentity/ase/bin/cli.sh status -u admin -p
Status: starting
```

After all cluster nodes are added, use the management or first node to carry out all cluster operations.

> **Note**
>
> Add one node at a time to the cluster. After the node completes loading data, add the next node.

Cluster nodes must be added sequentially, one node at a time, to ensure consistent cluster behavior. The following table lists the items that are synchronized across the cluster.

Item	Synchronized (Yes or No)	Synchronization (Restart or Live)
Certificates (keystore)	Yes	Restart
Master key	No	-
API JSON	Yes	Live and Restart

Copyright © 2025 Ping Identity Corporation

Item	Synchronized (Yes or No)	Synchronization (Restart or Live)
Cookies	Yes	Live and Restart
CLI admin password	No	No
Authorization token for sideband ASE	Yes	Live and Restart
Deny list and allow list (create, delete, and delete all)	Yes	Live and Restart
Real-time attacks (IP, cookie, and token is blocked)	Yes	Live
ase.conf	Yes	Restart
abs.conf	Yes	Restart
CLI commands that are *not* synchronized	The following commands are not synchronized: • `create_key_pair` • `create_csr` • `create_self_sign_cert` • `import_key_pair` • `import_cert` • `create_management_key_pair` • `create_management_csr` • `create_management_self_sign_cert` • `import_management_key_pair` • `import_management_cert` • `update_password` • `update_auth_method` • `generate_obfkey` • `obfuscate_keys` • `update_keystore_password` **Note** The commands listed above require the entire ASE to restart for the commands to synchronize.	-

Setting up an ASE cluster

Complete the following steps to setup an API Security Enforcer (ASE) cluster.

Copyright © 2025 Ping Identity Corporation

Before you begin

You must:

1. Obtain a list of Internet Protocol (IP) addresses and ports required for ASE cluster nodes.
2. Enable Network Time Protocol (NTP) on your system.
3. Back up the ASE data if you're adding an existing ASE instance to a cluster.

> **Note**
>
> When a node is added to a cluster, it synchronizes the data from the other nodes and overwrites existing data.

About this task

The following diagram provides an overview of the basic steps to setup and start an ASE cluster.

To setup an ASE cluster node:

Steps

1. Go to the `config` directory.
2. Edit the `ase.conf` file:
    1. Set `enable_cluster=true` for all cluster nodes.
    2. Make sure that the value in the parameter `mode` is the same on each ASE cluster node, either `inline` or `sideband`.

        > **Note**
        >
        > If the value of mode parameter does not match, the nodes will not form a cluster.

3. Edit the `cluster.conf` file:
    1. Configure `cluster_id` with an identical value for all nodes in a single cluster.

        Example:

        `[.parmname]` cluster_id= `[.option]` shopping````

    2. Enter the port number in the `cluster_management_port` parameter.

        The ASE node uses this port number to communicate with other nodes in the cluster.. The default port is 8020.

    3. Enter an IPv4 address or host name with the port number for the `peer_node`, which is the first (or any existing) node in the cluster.

Copyright © 2025 Ping Identity Corporation

> **Important**
>
> Keep this parameter empty for the first node of the cluster.

4. Provide the obfuscated `cluster_secret_key`.

   All the nodes of the cluster must have the same obfuscated `cluster_secret_key`. You must enter this key manually on each node of the cluster for the nodes to connect to each other.

5. For the first node of the ASE cluster, `peer_node` should be left empty. On other nodes of the ASE cluster, enter the IP address or the host name of the first cluster in the node in the `peer_node` variable.

   Example:

   The following is a sample `cluster.conf` file:

```
; API Security Enforcer's cluster configuration.
; This file is in the standard .ini format. The comments start with a semicolon (;).
; Section is enclosed in []
; Following configurations are applicable only if cluster is enabled with true in ase.conf
; unique cluster id.
; valid character class is [ A-Z a-z 0-9 _ - . / ]
; nodes in same cluster should share same cluster id
cluster_id=ase_cluster
; cluster management port.
cluster_manager_port=8020
; cluster peer nodes.
; a comma-separated list of hostname:cluster_manager_port or IPv4_address:cluster_manager_port
; this node will try to connect all the nodes in this list
; they should share same cluster id
peer_node=
; cluster secret key.
; maximum length of secret key is 128 characters (deobfuscated length).
; every node should have same secret key to join same cluster.
; this field cannot be empty.
; change default key for production.
cluster_secret_key=OBF:AES:nPJOh3wXQWK/BOHrtKu3G2SGiAEElOSvOFYEiWfIVSdummoFwSR8rDh2bBnhTDdJ:
7LFcqXQlqkW9kldQoFg0nJoLSojnzHDbD3iAy84pT84
```

4. After configuring an ASE node, start the node by running the following command:

```
/opt/pingidentity/ase/bin/start.sh
```

Scaling up the ASE cluster

Scale up the API Security Enforcer (ASE) cluster by adding one node at a time to an active cluster without disrupting traffic.

About this task

To add a new cluster node:

Copyright © 2025 Ping Identity Corporation

Steps

1. Enter the `peer_node` IP address or host name in the `cluster.conf` file of the ASE node.

   Example:

   If the IP of the first node is 192.168.20.121 with port 8020, then the `peer_node` parameter would be 192.168.20.121:8020.

   ```
   ; ASE cluster configuration. These configurations apply only when you have enabled cluster in the
   api_config file.
   ; Unique cluster ID for each cluster. All the nodes in the same cluster should have the same cluster
   ID.
   cluster_id=ase_cluster
   ; Cluster management port.
   cluster_manager_port=8020
   ; Cluster's active nodes. This can be a comma separated list of nodes in
   ipv4_address:cluster_manager_port format.
   peer_node=192.168.20.121:8020
   ```

2. Start the ASE node by changing the working directory to `bin` and running the `start.sh` script.

   Result:

   The new node synchronizes configuration and cookie data from the peer nodes. After loading, it becomes part of the cluster.

Scaling down the ASE cluster

You can remove a node from an active API Security Enforcer (ASE) cluster without disrupting traffic.

Steps

1. Stop the ASE node to be removed by changing the working directory to `bin` and running the `stop.sh` command.
2. Set the `enable_cluster` option as `false` in its `ase.conf` file.

> **Note**
>
> The removed node retains the cookie and certificate data from when it was part of the cluster.

Deleting an ASE cluster node

You can delete an inactive API Security Enforcer (ASE) cluster node that has either become unreachable or has been stopped.

About this task

To identify which cluster nodes are inactive and should be removed:

Steps

1. Run the `cluster_info` command.

Copyright © 2025 Ping Identity Corporation

Example:

```
/opt/pingidentity/ase/bin/cli.sh cluster_info -u admin -p
cluster id : ase_cluster
cluster nodes
127.0.0.1:8020 active
Step 1.1.1.1:8020 active
Step 2.2.2.2:8020 inactive
172.17.0.4:8020(tasks.aseservice) active
172.17.0.5:8020(tasks.aseservice) inactive
tasks.aseservice2:8020 not resolved
```

2. To delete the inactive node, run the `delete_cluster_node` command:

```
/opt/pingidentity/ase/bin/cli.sh delete_cluster_node  <IP:Port>
```

Example:

From the previous example, to remove inactive cluster nodes 2.2.2.2:8020 and 172.17.05:8020, it would look like the following:

```
/opt/pingidentity/ase/bin/cli.sh delete_cluster_node 2.2.2.2:8020
/opt/pingidentity/ase/bin/cli.sh delete_cluster_node 172.17.05:8020
```

Stopping the ASE cluster

Stop the API Security Enforcer (ASE) cluster.

About this task

To stop the entire cluster:

Steps

- Run the following command on any ASE node in the cluster:

  Example:

```
/opt/pingidentity/ase/bin/stop.sh cluster –u admin –p
```

Result

When the cluster stops, each cluster node retains all the cookie and certificate data.

Configuring the ASE cluster SSL

API Security Enforcer (ASE) supports Secure Sockets Layer (SSL) over TCP for securing communications between nodes in a cluster. It uses TLS 1.2 to encrypt the communications. You can view cluster information in the `controller.log` file available in the `/<pi_install path>/pingidentity/ase/logs/` directory.

Copyright © 2025 Ping Identity Corporation

About this task

Configure SSL in ASE using one of the following three methods:

Steps

1. Using the default certificate
2. Securing an ASE cluster
3. Importing an existing certificate and key pair

Using the default certificate

ASE ships with its default PKCS#12 key store located at `/<pi_install_path>/pingidentity/ase/config/cert/ase.store`. The default certificate and SSL keys are stored in the PKCS store. You can use them to secure the ASE cluster.

About this task

To synchronize the Secure Sockets Layer (SSL) certificate and keys across different nodes in an ASE cluster:

Steps

1. Start the ASE cluster.

   For more information on starting the ASE cluster, see Setting up an ASE cluster.

   > **Note**
   >
   > During the cluster start, cluster keys and certificates are synchronized across all the ASE nodes.

2. After the cluster is started, restart the secondary nodes of the cluster for the changes to take effect.

   For more information on restarting the cluster, see Restarting an ASE cluster.

Securing an ASE cluster

You can secure an API Security Enforcer (ASE) cluster using a new SSL certificate.

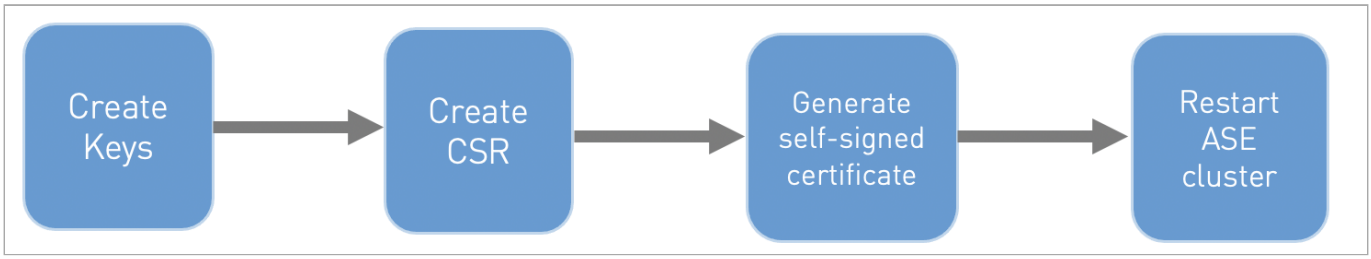
To achieve this, you can either use a self-signed certificate or a certificate authority (CA)-signed SSL certificate:

Copyright © 2025 Ping Identity Corporation

Self-signed certificate

Creating a self-signed certificate

About this task



To create a self-signed certificate:

Steps

1. Create a cluster key pair by running the following command-line interface (CLI) command:

```
create_cluster_key_pair [--yes | -y]
create private key for cluster server
--yes | -y : create private key without confirmation prompt
```

> **Note**
>
> The Private key in the pair is automatically created and updated in the keystore in `<pi_install_path>/pingidentity/ase/config/certs/` directory.

Example:

The following command creates `dh1024.pem` in the `/opt/pingidentity/ase/config/certs/cluster/` directory:

```
$ pingidentity/ase/bin/cli.sh -u admin -p admin create_cluster_key_pair
Warning: create_cluster_key_pair will delete any existing cluster key_pair, CSR and self-signed
certificate
Do you want to proceed [y/n]:y
Ok, creating new cluster key pair. Creating DH parameter may take around 20 minutes. Please wait
Cluster key created at keystore
Cluster dh param file created at /opt/pingidentity/ase/config/certs/cluster/dh1024.pem
```

2. Generate a certificate signing request (CSR) from the private key running the following CLI command:

```
create_cluster_csr [--yes | -y]
create certificate signing request for cluster server
--yes | -y : create certificate signing request without confirmation prompt
```

Example:

Copyright © 2025 Ping Identity Corporation

The following command creates a `.csr` file in the `/opt/pingidentity/ase/config/certs/cluster/` directory:

```
$ pingidentity/ase/bin/cli.sh -u admin -p admin create_cluster_csr
Warning: create_cluster_csr will delete any existing cluster CSR and self signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >OP
State >GP
Location >IP
Organization >KP
Organization Unit >Kpase
Common Name >www.123.com
Generating CSR. Please wait...
OK, cluster csr created at /opt/pingidentity/ase/config/certs/cluster/cluster.csr
```

Result:

This `.csr` file is saved in the `<pi_install_path>/pingidentity/ase/config/certs/cluster/` directory.

3. Generate a self-signed certificate by running the following command:

```
create_cluster_self_sign_cert [--yes | -y]
--yes | -y : create self signed certificate without confirmation prompt
```

Example:

The following command creates a self-signed certificate in the key store:

```
$ pingidentity/ase/bin/cli.sh -u admin -p admin create_cluster_self_sign_cert
Warning: create_cluster_self_sign_cert will delete any existing cluster self signed certificate
Do you want to proceed [y/n]:y
Creating new cluster self signed certificate
OK, self sign certificate created in key store
```

> **Note**
>
> The certificate is automatically created in the key store in the `<pi_install_path>/pingidentity/ase/config/certs/` directory.

4. Restart the ASE cluster for synchronizing the key and certificate.

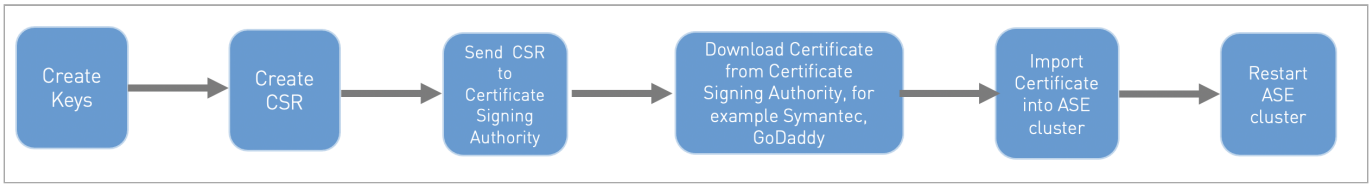
> **Note**
>
> For more information, follow the instructions in Restarting an ASE cluster.

Copyright © 2025 Ping Identity Corporation

CA-signed certificate

Creating a CA-signed certificate

About this task



To create a CA-signed SSL certificate:

Steps

1. **Create a cluster key pair by running the following CLI command:**

```
create_cluster_key_pair [--yes | -y]
create private key for cluster server
--yes | -y : create private key without confirmation prompt
```

Example:

The following command creates a key in the `/opt/pingidentity/ase/config/certs/cluster/` directory:

```
$ pingidentity/ase/bin/cli.sh -u admin -p admin create_cluster_key_pair
Warning: create_cluster_key_pair will delete any existing cluster key_pair, CSR and self-signed
certificate
Do you want to proceed [y/n]:y
Ok, creating new cluster key pair. Creating DH parameter may take around 20 minutes. Please wait
Cluster key created at keystore
Cluster dh param file created at /opt/pingidentity/ase/config/certs/cluster/dh1024.pem
```

> **Note**
>
> The private key in the pair is automatically created and updated in the key store in the `<pi_install_path>/pingidentity/ase/config/certs/` directory.

2. **Generate a certificate signing request (CSR) from the private key using the following CLI command:**

```
create_cluster_csr [--yes | -y]
create certificate signing request for cluster server
--yes | -y : create certificate signing request without confirmation prompt
```

> **Note**
>
> This `.csr` file gets saved in the `<pi_install_path>/pingidentity/ase/config/certs/cluster/` directory.

Example:

Copyright © 2025 Ping Identity Corporation

The following command creates a `.csr` file in the `/opt/pingidentity/ase/config/certs/cluster/` directory:

```
$ pingidentity/ase/bin/cli.sh -u admin -p admin create_cluster_csr
Warning: create_cluster_csr will delete any existing cluster CSR and self signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >OP
State >GP
Location >IP
Organization >KP
Organization Unit >Kpase
Common Name >www.123.com
Generating CSR. Please wait...
OK, cluster csr created at /opt/pingidentity/ase/config/certs/cluster/cluster.csr
```

3. Upload the CSR created in step 2 to the CA-signing authority's website to get a CA-signed certificate.
4. Download the CA-signed certificate from the CA-signing authority's website.
5. Import the signed CA-certificate into ASE cluster by running the following CLI command:

```
import_cluster_cert {cert_path} [--yes | -y]
import CA signed certificate for cluster server
--yes | -y : import CA signed certificate without confirmation prompt
```

> **Note**
>
> The certificate is imported into the key store in the `<pi_install_path>/pingidentity/ase/config/certs/` directory.

Example:

```
./cli.sh -uadmin -padmin import_cluster_key_pair /home/ec2-user/cert_folder/signed_cert/
test.elasticbeam.com.key
Warning: import_cluster_key_pair will overwrite any existing cluster certificates
Do you want to proceed [y/n]:y
Exporting cluster key to API Security Enforcer...
OK, key pair added to keystore
2:43
[ec2-user@rhel76-cluster-nodes-6-12 bin]$ ./cli.sh -uadmin -padmin import_cluster_cert /home/
ec2-user/cert_folder/signed_cert/test.elastic.crt
Warning: import_cluster_cert will overwrite any existing cluster signed certificate
Do you want to proceed [y/n]:y
Exporting cluster certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

6. Synchronizing the key and certificate by restarting the ASE cluster.

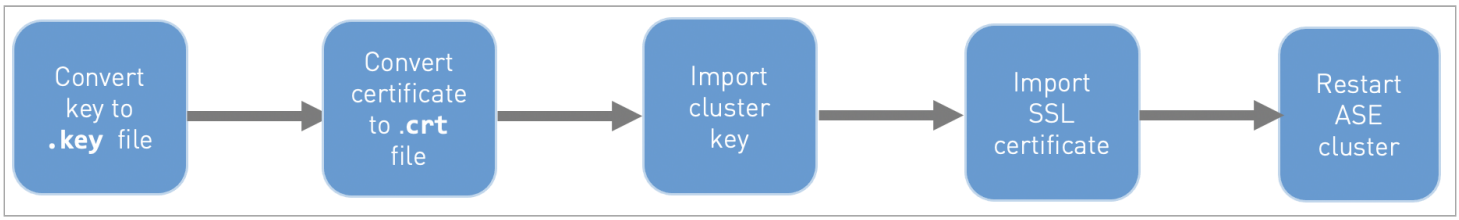
> **Note**
>
> For more information, follow the instructions in Restarting an ASE cluster.

Copyright © 2025 Ping Identity Corporation

Importing an existing certificate and key pair

About this task

The following diagram shows an overview of the steps for importing an existing certificate and key pair.



To import an existing certificate and key pair:

Steps

1. Convert the key to a `.key` file:

   ```
   openssl rsa -in private.pem -out private.key
   ```

2. Convert the SSL certificate to a `.crt` file:

   ```
   openssl x509 -in server-cert.pem -out server-cert.crt
   ```

3. Import the cluster key into the key store using the following CLI command.

   ```
   import_key_pair {key_path} [--yes | -y]
   import key pair for cluster server
   --yes | -y : import key pair without confirmation prompt
   ```

4. Import the certificate into the key store using following CLI command:

   ```
   import_cert {cert_path} [--yes | -y]
   import CA signed certificate for cluster server
   --yes | -y : import CA signed certificate without confirmation prompt
   ```

5. Restart the API Security Enforcer (ASE) cluster for synchronizing the key and the certificate.

   For more information on restarting the ASE cluster, see Restarting an ASE cluster.

Copyright © 2025 Ping Identity Corporation

Restarting an ASE cluster

To ensure consistent cluster behavior, you should restart API Security Enforcer (ASE) cluster nodes, one node at a time.

Steps

1. Stop all the nodes in the cluster by running the following command on any ASE node in the cluster:

```
/opt/pingidentity/ase/bin/stop.sh cluster –u admin –p
```

2. Start the first node or management node in the cluster by running the following command:

```
/opt/pingidentity/ase/bin/start.sh
```

> **Note**
>
> The first node or management node of the ASE cluster has the `peer_node` parameter empty in the `cluster.conf` file.

3. Verify the status of the node by running the `status` command.
4. Start the next node in the cluster only after the status of the node changes to `started`.

   Example:

```
/opt/pingidentity/ase/bin/cli.sh status -u admin -p
Status: started
```

5. Repeat steps 2 and 3 for all the other nodes in the cluster to complete the cluster restart.

API JSON files configuration

Learn what application programming interface (API) JavaScript Object Notation (JSON) files are and how they are configured to secure the APIs in your environment.

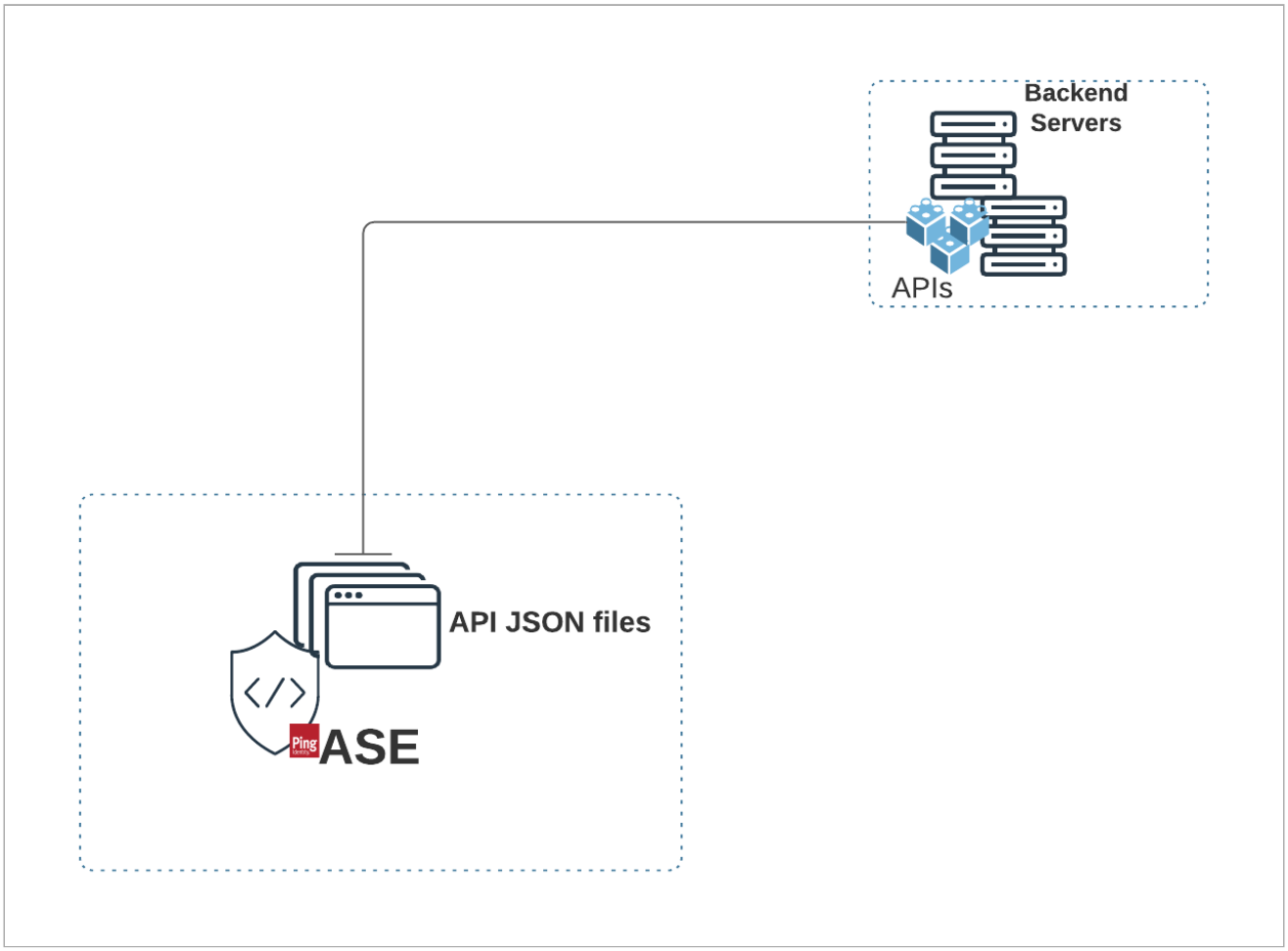
API JSON files are used to configure the behavior and properties of your APIs in API Security Enforcer (ASE). The parameters in API JSON files help ASE to uniquely identify the APIs in your environment. Each API has a unique API JSON file in ASE. ASE ships with sample JSON files located in the `/config/api` directory.

The parameters configured in an API JSON file help ASE extract metadata from API traffic, set decoys to trap intruding attacks, perform health checks on backend servers, and so on. The API JSON parameters also help the API Behavioral Security (ABS) AI Engine to build AI models to detect any indicators of attacks (IoAs) on APIs.

See the following for more information on the parameters in API JSON files:

- Defining an API using API JSON configuration file in sideband mode
- Defining an API using API JSON configuration file in inline mode

Copyright © 2025 Ping Identity Corporation



You can manually configure the JSON file with the required parameters and add them to ASE.

> **Note**
>
> The sample JSON file has an extension of `.example`. If you are customizing the example file, then save the file as a `.json` file.

Adding API JSON files

About this task

After configuring an API JSON file, add it to ASE to activate ASE processing.

Steps

- To add an API, do one of the following:

  Choose from:

  - Run the following command-line interface (CLI) command:

```
/<ASE_Installation path>/pingidentity/ase/bin/cli.sh –u admin -p admin add_api {file_path/
api_name}
```


Copyright © 2025 Ping Identity Corporation

- Use the Create API in ASE Admin APIs to add an API JSON file to ASE.

  The following is a sample `curl` command for it:

```
curl --location --request POST '{{API}}=<API Name>' \
--header '{{Access_Key_Header}}: {{Access_Key}}' \
--header '{{Secret_Key_Header}}: {{Secret_key}}' \
--header 'Content-Type: application/json' \
--data-raw '{
    "api_metadata": {
        "protocol": "https",
        "url": "/patmapp",
        "hostname": "*",
        "oauth2_access_token": false,
        "apikey_qs": "",
       <<Request body continues...>>
```

Listing API JSON files

Steps

- Check the addition of an API JSON file to ASE by running the following CLI command:

```
/<ASE_Installation path>/pingidentity/ase/bin/cli.sh –u admin -p admin list_api
```

- Use List API in ASE Admin APIs to verify.

  Example:

  The following is a sample `curl` command for it:

```
curl --location --request GET '{{List_API}}' \
--header '{{Access_Key_Header}}: {{Access_Key}}' \
--header '{{Secret_Key_Header}}: {{Secret_key}}'
```

Updating API JSON files

About this task

After activation, an API JSON definition can be updated in real time.

Steps

- To update the API JSON file:

  Choose from:

**

1. Edit the API JSON file located in the `/config/api` directory and make the desired changes.
2. Save the edited API JSON file and run the following CLI command:

Copyright © 2025 Ping Identity Corporation

```
/<ASE_Installation path>/pingidentity/ase/bin/cli.sh –u admin -p admin update_api  <api_name>
```

The following is an example:

```
/opt/pingidentity/ase/bin/cli.sh –u admin -p admin update_api shop
api shop updated successfully
```

- You can also use Update API in ASE Admin APIs to update the JSON.

The following is a sample `curl` command for it:

+

```
curl --location --request PUT '{{API}}=<API Name>' \
--header '{{Access_Key_Header}}: {{Access_Key}}' \
--header '{{Secret_Key_Header}}: {{Secret_key}}' \
--header 'Content-Type: application/json' \
--data-raw '{
    "api_metadata": {
        "protocol": "https",
        "url": "/pubatmapp",
        "hostname": "*",
        "oauth2_access_token": false,
        <<Request body continues...>>
```

Configuring SSL for external APIs

API Security Enforcer (ASE) supports both TLS 1.2 and Secure Sockets Layer (SSL) 3 for external application programming interface (API)s.

OpenSSL is bundled with ASE. The following are the version details:

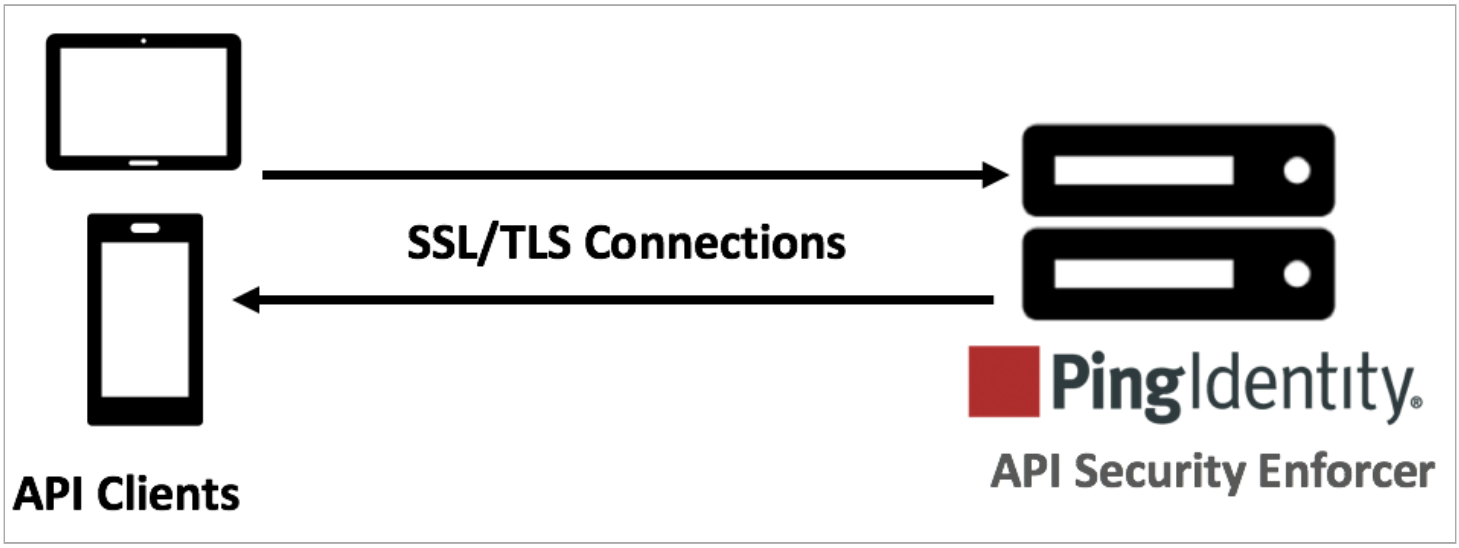
- RHEL 7 : OpenSSL 1.0.2k-fips January 26, 2017
- Ubuntu 16LTS : OpenSSL 1.0.2g March 1, 2016

You can configure SSL in ASE for client side connection using one of the following methods:

- Using a certificate authority (CA)-signed certificate
- Using a self-signed certificate
- Using an existing certificate

The steps provided in this section are for certificate and key generated for connections between the client and ASE as depicted in the following diagram.

Copyright © 2025 Ping Identity Corporation



In a cluster setup:

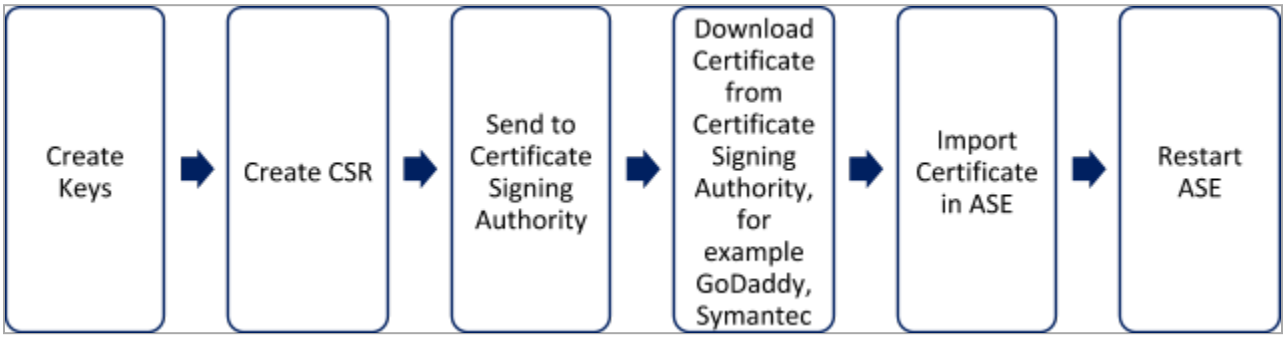
1. Stop all the ASE cluster nodes.
2. Configure the certificate on the management node.
3. Start the cluster nodes one by one for the certificates to synchronize across the nodes.

Copyright © 2025 Ping Identity Corporation

CA-signed certificate

Using a CA-signed certificate
About this task

To use a CA-signed SSL certificates, follow the process to create a private key, generate a certificate signing request (CSR), and request a certificate as shown in the following diagram.



> **Note**
>
> ASE internally validates the authenticity of the imported certificate.

To use a CA-signed certificate:

Steps

1. Create a private key.

   The ASE command-line interface (CLI) is used to create a 2048-bit private key and to store it in the key store.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh create_key_pair -u admin -p
   Warning: create_key_pair will delete any existing key_pair, CSR and self-signed certificate
   Do you want to proceed [y/n]:y
   Ok, creating new key pair. Creating DH parameter may take around 20 minutes. Please wait
   Key created in keystore
   dh param file created at /opt/pingidentity/ase/config/certs/dataplane/dh1024.pem
   ```

2. Create a CSR.

   ASE takes you through a CLI-based interactive session to create a CSR.

Copyright © 2025 Ping Identity Corporation

Example:

```
/opt/pingidentity/ase/bin/cli.sh create_csr -u admin -p
Warning: create_csr will delete any existing CSR and self-signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >US
State > Colorado
Location >Denver
Organization >Pingidentity
Organization Unit >Pingintelligence
Common Name >ase
Generating CSR. Please wait...
OK, csr created at /opt/pingidentity/ase/config/certs/dataplane/ase.csr
```

3. Upload the CSR that you created in step 2 to the CA signing authority's website to get a CA-signed certificate.
4. Download the CA-signed certificate from the CA signing authority's website.
5. Use the CLI to import the signed CA certificate into ASE.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh import_cert  <CA signed certificate path>  -u admin -p
   Warning: import_cert will overwrite any existing signed certificate
   Do you want to proceed [y/n]:y
   Exporting certificate to API Security Enforcer...
   OK, signed certificate added to keystore
   ```

   Result:

   The certificate is imported into the key store.

6. Restart ASE.

   For more information, see Starting and stopping ASE.

Copyright © 2025 Ping Identity Corporation

Self-signed certificate

Using a self-signed certificate
About this task

> **Note**
> A self-signed certificate is also supported for customer testing.

To create a self-signed certificate:

Steps

1. Create a private key.

   ASE CLI is used to generate a 2048-bit private key in the `/opt/pingidentity/ase/config/certs/dataplane/dh1024.pem` file.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh create_key_pair -u admin -p
   Warning: create_key_pair will delete any existing key_pair, CSR and self-signed certificate
   Do you want to proceed [y/n]:y
   Ok, creating new key pair. Creating DH parameter may take around 20 minutes. Please wait
   Key created in keystore
   dh param file created at /opt/pingidentity/ase/config/certs/dataplane/dh1024.pem
   ```

2. Create a CSR file.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh create_csr -u admin -p
   Warning: create_csr will delete any existing CSR and self-signed certificate
   Do you want to proceed [y/n]:y
   please provide following info
   Country Code >US
   State >colorado
   Location >Denver
   Organization >PI
   Organization Unit >TEST
   Common Name >yoursiteabc.com
   Generating CSR. Please wait...
   OK, csr created at /opt/pingidentity/ase/config/certs/dataplane/ase.csr
   ```

3. Use the CLI to produce a self-signed certificate using the certificate request located in `/pingidentity/ase/config/certs/dataplane/ase.csr`.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh create_self_sign_cert -u admin -p
   Warning: create_self_sign_cert will delete any existing self-signed certificate
   Do you want to proceed [y/n]:y
   Creating new self-signed certificate
   OK, self-sign certificate created in keystore
   ```


Copyright © 2025 Ping Identity Corporation

4. Restart ASE.

   For more information, see Starting and stopping ASE.

Existing certificate and key pair

Using an existing certificate and key pair
About this task

> **Note**
> If you have an intermediate certificate from a CA, then append the content to your server `.crt` file.

To install an existing certificate:

Steps

1. Import the key pair.

   Example:

```
/opt/pingidentity/ase/bin/cli.sh import_key_pair private.key -u admin -p
Warning: import_key_pair will overwrite any existing certificates
Do you want to proceed [y/n]:y
Exporting key to API Security Enforcer...
OK, key pair added to keystore
```

2. Import the `.crt` file in ASE by running the `import_cert` CLI command.

   Example:

```
/opt/pingidentity/ase/bin/cli.sh import_cert server-crt.crt -u admin -p
Warning: import_cert will overwrite any existing signed certificate
Do you want to proceed [y/n]:y
Exporting certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

3. Restart ASE.

   For more information, see Starting and stopping ASE.

Configuring SSL for management APIs

API Security Enforcer (ASE) supports both TLS 1.2 and Secure Sockets Layer (SSL)3 for management application programming interface (API)s.

Copyright © 2025 Ping Identity Corporation

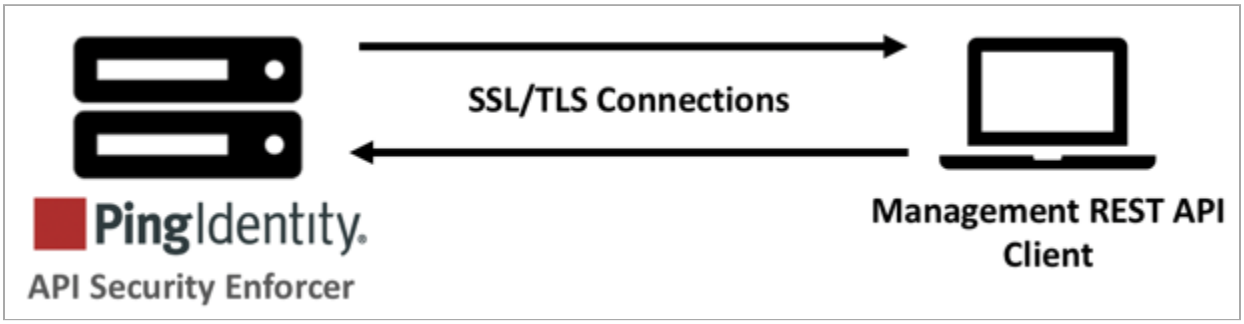
OpenSSL is bundled with ASE. The following are the version details:

- RHEL 7 : OpenSSL 1.0.2k-fips 26 Jan 2017
- Ubuntu 16LTS: OpenSSL 1.0.2g 1 Mar 2016

You can configure SSL in ASE for management APIs using one of the following methods:

- Using a Certificate Authority (CA)-signed certificate
- Using a self-signed certificate
- Using an existing certificate

The steps provided in this section are for certificate and key generated are for connections between a management API client and ASE:

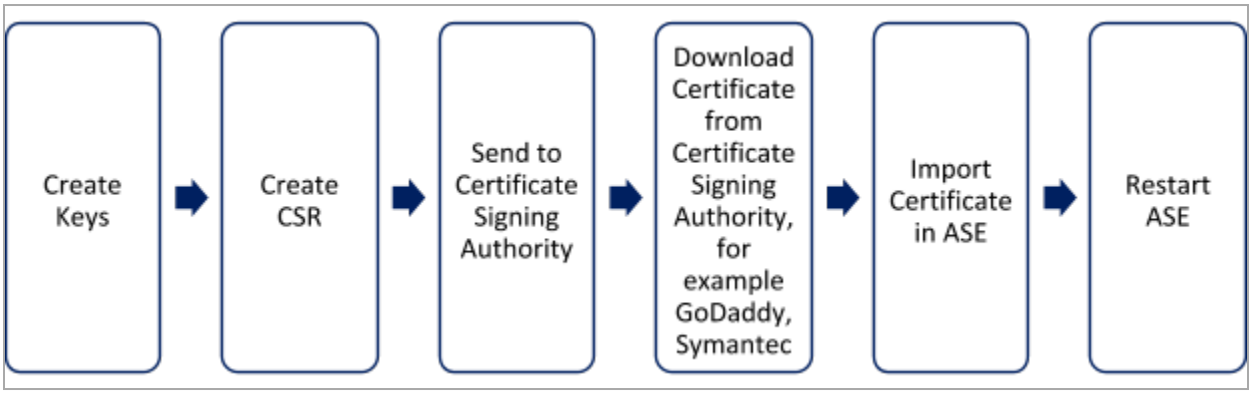


In a cluster setup:

1. Stop all the ASE cluster nodes.
2. Configure the certificate on the management node.
3. Start the cluster nodes one by one for the certificates to synchronize across the nodes.

Using a CA-signed certificate

To use Certificate Authority (CA) signed SSL certificates, follow the process to create a private key, generate a Certificate Signing Request (CSR), and request a certificate as shown below:



Copyright © 2025 Ping Identity Corporation

> **Note**
>
> ASE internally validates the authenticity of the imported certificate.

To use a CA-signed certificate:

1. Create a private key. ASE command-line interface (CLI) is used to create a 2048-bit private key and to store it in the `/opt/pingidentity/ase/config/certs/management` directory.

```
/opt/pingidentity/ase/bin/cli.sh create_management_key_pair -u admin -p
Warning: create_management_key_pair will delete any existing management key_pair, CSR and self-signed certificate
Do you want to proceed [y/n]:y
Ok, creating new management key pair. Creating DH parameter may take around 20 minutes. Please wait
Management key created at keystore
Management dh param file created at /opt/pingidentity/ase/config/certs/management/dh1024.pem
```

2. Create a CSR. ASE takes you through a CLI-based interactive session to create a CSR.

```
/opt/pingidentity/ase/bin/cli.sh create_management_csr -u admin -p
Warning: create_management_csr will delete any existing management CSR and self-signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >US
State >Colorado
Location >Denver
Organization >Pingidentity
Organization Unit >Pingintelligence
Common Name >management.ase
Generating CSR. Please wait...
OK, management csr created at /opt/pingidentity/ase/config/certs/management/management.csr
```

3. Upload the CSR created in step 2 to the CA signing authority's website to get a CA signed certificate.
4. Download the CA-signed certificate from the CA signing authority's website.
5. Use the CLI to import the signed CA certificate into ASE. The certificate is imported into the `/pingidentity/config/certs/management/management.csr` file

```
/opt/pingidentity/ase/bin/cli.sh import_management_cert <CA signed certificate path> -u admin -p
Warning: import_management_cert will overwrite any existing management signed certificate
Do you want to proceed [y/n]:y
Exporting management certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

6. Restart ASE by first stopping and then starting ASE.

Using a self-signed certificate

A self-signed certificate is also supported for customer testing.

Copyright © 2025 Ping Identity Corporation

To create a self-signed certificate

1. Create a private key. ASE CLI is used to generate a 2048-bit private key which is in the `/ase/config/certs/` directory.

```
/opt/pingidentity/ase/bin/cli.sh create_management_key_pair -u admin -p
Warning: create_management_key_pair will delete any existing management key_pair, CSR and self-
signed certificate
Do you want to proceed [y/n]:y
Ok, creating new management key pair. Creating DH parameter may take around 20 minutes. Please wait
Management key created at keystore
Management dh param file created at /opt/pingidentity/ase/config/certs/management/dh1024.pem
```

2. Create a CSR. Enter the following command.

```
/opt/pingidentity/ase/bin/cli.sh create_management_csr -u admin -p
password >
Warning: create_csr will delete any existing CSR and self signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >US
State >colorado
Location >Denver
Organization >PingIdentity
Organization Unit >PI
Common Name >yoursiteabc.com

Generating CSR. Please wait...
OK, csr created at /opt/pingidentity/ase/config/certs/management/ase.csr
```

3. Create a self-signed certificate. Use the CLI to produce a self-signed certificate using the certificate request located in `/pingidentity/ase/config/certs/management/ase.csr`

```
/opt/pingidentity/ase/bin/cli.sh create_management_self_sign_cert -u admin -p
Warning: create_management_self_sign_cert will delete any existing management self-signed
certificate
Do you want to proceed [y/n]:y
Creating new management self-signed certificate
OK, self-sign certificate created in key store
```

4. Restart ASE by stopping and starting.

Importing an existing certificate and key pair

To install an existing certificate, complete the following steps and import it into ASE. If you have intermediate certificate from CA, then append the content to your server `.crt` file.

1. Convert the key from the existing `.pem` file:

```
openssl rsa -in private.pem -out private.key
```


Copyright © 2025 Ping Identity Corporation

2. Convert the existing `.pem` file to a `.crt` file:

```
openssl x509 -in server-cert.pem -out server-cert.crt
```

3. Import key pair from step 2:

```
/opt/pingidentity/ase/bin/cli.sh import_management_key_pair private.key -u admin -p
Warning: import_key_pair will overwrite any existing certificates
Do you want to proceed [y/n]:y
Exporting management key to API Security Enforcer...
OK, key pair added to keystore
```

4. Import the `.crt` file in ASE using the `import_management_cert` CLI command:

```
/opt/pingidentity/ase/bin/cli.sh import_management_cert server-crt.crt -u admin -p
Warning: import_management_cert will overwrite any existing management signed certificate
Do you want to proceed [y/n]:y
Exporting management certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

5. Restart ASE by stopping and starting.

Native and Pluggable Authentication Modules (PAM) authentication

API Security Enforcer (ASE) provides two types of authentication:

- Linux Pluggable Authentication Module (PAM)
- ASE native authentication (default method)

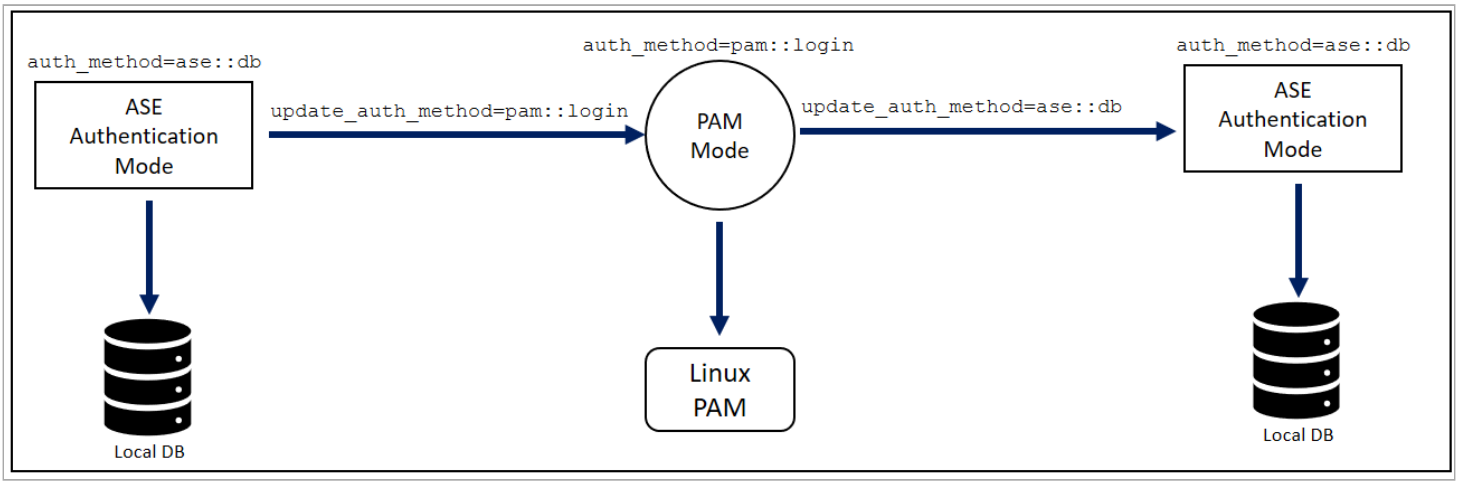
All actions carried out on ASE require an authenticated user.

The two methods to choose the authentication method include:

- Configure `auth_method` parameter in the `ase.conf` file. For more information, see ASE Initial Configuration.
- Run a command-line interface (CLI) command (`update_auth_method <method>`).

The following diagram shows the transition between authentication modes.

Copyright © 2025 Ping Identity Corporation



Note

The authentication method can be changed during run-time without restarting ASE.

Copyright © 2025 Ping Identity Corporation

Native authentication

Configuring ASE native authentication
About this task

By default, ASE uses native ASE authentication which ships with the system. Each user can run CLI commands by including the shared username and password with each command. The system ships with a default username (`admin`) and password (`admin`).

> **Important**
>
> Always change the default password using the **`update_password`** command. For more information on ASE commands, see Appendix A.

Steps

- To configure `ase.conf` to support native authentication, use the default configuration values:

```
auth_method=ase::db
```

- To change the authentication from native authentication to PAM mode, enter the following command in ASE command line:

> **Note**
>
> In the example, `login` is a PAM script used for authentication.

```
/opt/pingidentity/ase/bin/cli.sh update_auth_method pam::login -u admin -p
 <password>
```

- To switch from PAM mode authentication back to native authentication, issue the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh update_auth_method ase::db -u  <pam_user>  -p
 <password>
```

Example:

The following is an example of a CLI command with native authentication (-u,-p) enabled:

```
/opt/pingidentity/ase/bin/cli.sh add_server -u admin -p
 <password>
```

Copyright © 2025 Ping Identity Corporation

PAM authentication

Configuring Linux PAM authentication
About this task

PAM-based authentication provides the flexibility to authenticate administrators using existing authentication servers, such as your organization's Lightweight Directory Access Protocol (LDAP) directory. When PAM authentication is active, ASE logs the identity of the user executing each CLI command. This provides a user-specific audit trail of administrative access to the ASE system.

Steps

- To activate PAM-based authentication, configure `auth_method` in `ase.conf` as `pam::<service>`, where `<service>` is the script that the PAM module reads to authenticate the users.

> **Note**
>
> Service scripts include `login`, `su`, `ldap`, etc. For example, the **`login`** script allows all system users administrative access to ASE.

- To support PAM authentication with the **`login`** script, update the `auth_method` configuration values in `ase.conf`:

```
auth_method=pam::login
```

Example:

The following is an example using the CLI to change from Native to PAM authentication with **`login`** script:

```
/opt/pingidentity/ase/bin/cli.sh update_auth_method pam::login -u admin -p
 <password>
```

> **Warning**
>
> Make sure that the script name provided for PAM-based authentication is the correct one. If a wrong file name is provided, ASE administrators are locked out of ASE.

- To write your own PAM module script, add a custom script, such as **`ldap`**, that defines PAM's behavior for user authentication to the `/etc/pam.d` directory.
- To set the authentication method and use the **`ldap`** script, run the following command:

```
/opt/pingidentity/ase/bin/cli.sh update_auth_method pam::ldap -u admin -p
 <password>
```

Example:

Copyright © 2025 Ping Identity Corporation

In the following example, the PAM module uses the organization's LDAP server to authenticate users.

```
root@localhost:/# cat /etc/pam.d/ldap
auth    sufficient    pam_ldap.so     # Authenticate with LDAP server.
#auth   sufficient    pam_permit.so   # Allow everyone. Pass-through mode.
#auth   sufficient    pam_deny.so     # Disallow everyone. Block all access.
```

Recovering ASE from unavailable pam.d script

About this task

When an invalid script name is entered while changing to PAM authentication, the PAM module defaults to `etc/pam.d/others` for authentication. This makes ASE inaccessible to administrators. If this happens, you must recover ASE.

To recover ASE:

Steps

1. Copy `etc/pam.d/login` to `etc/pam.d/other`.

   Result:

   ASE will use the credentials in `etc/pam.d/login` to authenticate administrators.

2. After signing back on to ASE, change the authentication method to use the correct file name.

   > **Note**
   >
   > Copying the contents of `etc/pam.d/login` to `etc/pam.d/other` does not require a restart of ASE or the host operating system.

ASE access, management, and audit logs

API Security Enforcer (ASE) generates three kinds of logs:

Access logs

Contain information about all application programming interface (API) traffic.

Management logs

Contain information about Controller and Balancer.

Audit logs

Contain information about various commands executed in ASE.

Related links

- Changing management log levels

Copyright © 2025 Ping Identity Corporation

- Purge log files
- Configuring a syslog server

Access

Access logs

Access logs are generated for port 80 (default port) and 443 (default port) traffic. Each Balancer process has a corresponding Access log file (that is. two port 80 Balancer processes and two port 443 Balancer processes require four log files). The log file name format is `<protocol><port>_pid<process-ID>access<date>.log`.

The following are examples for port 80 and port 443:

- `http_ws_80_pid_19017_access_2018-01-22_13-10.log`
- `https_wss_443_pid_19018_access_2018-01-22_13-10.log`

Access logs are rotated every 10 minutes and archived. The archived log file format has `.gz` at the end of the log file name, for example `http_ws_80_pid_19017_access_2018-01-22_13-10.log.gz`.

ASE sends all archived log files to API Behavioral Security (ABS) to detect attacks using machine learning algorithms. The files are then moved to the `logs/abs_uploaded` directory.

The following snippet shows an example log file:

```
-rw-r--r--. 1 root root 0 Aug 10 13:10 http_ws_80_pid_0access2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10 https_wss_443_pid_0access2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10 http_ws_80_pid_19010access2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10 http_ws_80_pid_19009access2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10 https_wss_443_pid_19022access2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10 https_wss_443_pid_19017access2018-01-22_13-10.log
-rw-r--r--. 1 root root 33223 Aug 10 13:11 balancer.log
-rw-r--r--. 1 root root 20445 Aug 10 13:11 controller.log
-rw-r--r--. 1 root root 33244 Aug 10 13:11 balancer_ssl.log
```


Copyright © 2025 Ping Identity Corporation

Management

Management logs

Management log detail levels, such as `INFO`, `WARNING`, and `DEBUG`, are configured in `ase.conf`.

Generated by controller and balancers, management logs are stored in the logs directory and include:

- Controller logs – `controller.log`
- Balancer log for port 80 (default port) – `balancer.log`
- Balancer log for port 443 – `balancer_ssl.log`

== Controller logs

The `controller.log` file is a log file with data from the command-line interface (CLI), REST API, configurations, IPC, SSL, cluster, and ABS. Rotated every 24 hours, `controller.log` is the current file name. Older files are appended with a timestamp.

== Balancer logs

The `balancer.log` file for port 80 and `balancer_ssl.log` file for port 443 are static files that are not rotated. These files contain information about IPC between controllers and balancer processes as well as IPC between balancer processes.

In a sideband ASE deployment, the balancer checks for request-response parsing error every 30 seconds. Parsing error statistics are logged in `balancer.log` file only if the balancer encounters parsing errors. If there are no errors in a 30-second period, the `balancer.log` file does not show the JSON output.

The following is a snippet of request-response parsing error statistics:

```
{
  "sideband stats": {
    "request parsing errors": {
      "total requests failed": 1,
      "request body absent": 0,
      "request body malformed": 0,
      "request source ip absent": 1,
      "request source ip invalid": 0,
      "request method absent": 0,
      "request url absent": 0,
      "request host header absent": 0,
      "request authentication failure": 0,
      "request error unknown": 0
    },
    "response parsing errors": {
      "total responses failed": 1,
      "response body absent": 0,
      "response body malformed": 0,
      "response code absent": 0,
      "response authentication failure": 0,
      "response correlation id not found": 1,
      "response error unknown": 0
    }
  }
}
```

Copyright © 2025 Ping Identity Corporation

The snippet shows that there was one parsing error for request and one for the response. The statistics also lists the type of request and response error.

Audit

Audit logs

ASE logs administrator actions, such as CLI commands and configuration changes, and stores audit logs in the `opt/pingidentity/ase/logs` directory. Performed on a per ASE node basis, audit logging is enabled by default.

Use the CLI to enable or disable audit logging using the commands `enable_audit` and `disable_audit`.

For example, to enable audit logs, enter the following at the command line:

```
/opt/pingidentity/ase/bin/cli.sh enable_audit -u admin -p  <password>
```

The audit log captures information related to:

- System changes using CLI or REST API calls
- API JSON changes or `ase.conf` file updates
- SSL certificate updates

The logs are rotated every 24 hours with the current log file having no timestamp in its name. For more information, see Audit log.

The following is a snippet of audit log files:

```
-rw-r--r-- 1 root root 358 Aug 13 10:00 audit.log.2018-08-13_09-54
-rw-r--r-- 1 root root 301 Aug 13 10:12 audit.log.2018-08-13_10-00
-rw-r--r-- 1 root root 1677 Aug 13 11:16 audit.log.2018-08-13_10-12
-rw-r--r-- 1 root root 942 Aug 14 06:26 audit.log.2018-08-14_06-22
-rw-r--r-- 1 root root 541 Aug 15 08:19 audit.log
```

Changing management log levels

Complete the following steps to change management log levels.

About this task

The management log (`balancer.log` and `controller.log`) levels are initially configured in `ase.conf` file by setting `log_level` to one of the following five values:

- `fatal`

Copyright © 2025 Ping Identity Corporation

- `error`
- `warning`
- `info`
- `debug`

> **Note**
>
> The default value is `info`.

Steps

- To change the log level of management logs during run-time, use the `log_level` command.

  > **Note**
  >
  > The **`log_level`** command works in an identical way for both sideband and inline API Security Enforcer (ASE) modes.

- In an ASE cluster set up, run the `log_level` command with the `warn` option on all the ASE nodes in the command-line interface (CLI).

  Example:

  The following is an example CLI output of the `log_level` command to change the log level to `warning`:

  ```
  #./bin/cli.sh -u admin -p admin log_level warn
  ```

  Result:

  The change in log-level is also recorded in Audit logs.

- Verify the current log level by using the ASE `status` command.

  Example:

  ```
  #./bin/cli.sh -u admin -p status
  API Security Enforcer
  status              : started
  mode                : inline
  http/ws             : port 8080
  https/wss           : port 8443
  firewall            : enabled
  abs                 : disabled, ssl: enabled
  abs attack          : disabled
  audit               : enabled
  ase detected attack : disabled
  attack list memory  : configured 128.00 MB, used 25.60 MB, free 102.40 MB
  log level           : warn
  timezone            : local (MST)
  ```

Copyright © 2025 Ping Identity Corporation

Purge log files

To manage storage space, you can either archive or purge access log, controller log, and audit log files that have been uploaded to API Behavioral Security (ABS). API Security Enforcer (ASE) provides a `purge.sh` script to remove access log files from the `abs_uploaded` directory.

The `purge` script is part of the `/opt/pingidentity/ase/util` directory.

> **Warning**
>
> When the purge script is run, the access log files are permanently deleted from ASE.

To run the purge script, enter the following in the ASE command-line interface (CLI):

```
/opt/pingidentity/ase/util/purge.sh -d 3
In the above example, purge.sh deletes all the access log files which are older than 3 days. Here is a
sample output for the purge script.
admin@pingidentity# ./util/purge.sh -d 3
This will delete logs in /opt/pingidentity/ase/logs/abs_uploaded that is older than 3 days.
Are you sure (yes/no): yes
removing /opt/pingidentity/ase/logs/abs_uploaded/Processed_decoy_pid_278892017-04-01_11-04.log.gz : last
changed at Sat Apr 1 11:11:01 IST 2017
removing /opt/pingidentity/ase/logs/abs_uploaded/Processed_http_ws_80_pid_27905access__2017-04-01_11-04.log.gz :
last changed at Sat Apr 1 11:11:01 IST 2017
```

External log archival

The `purge` script can also archive logs to secondary storage for future reference. The purge script provides an option to choose the number of days to archive the log files. Use the `-l` option and the path of the secondary storage to place the archived log files. For example:

```
admin@pingidentity# ./util/purge.sh -d 3 -l /tmp/
```

In the above example, log files older than three days are archived to the `tmp` directory. To automate log archival, add the script to a cron job.

Related links

- Changing management log levels
- ASE access, management, and audit logs
- Configuring a syslog server

Configuring a syslog server

Syslog messages are a standard for sending event notification messages. These messages can be stored locally or on an external syslog server. API Security Enforcer (ASE) generates and sends syslog messages to an external syslog server over UDP. All the syslog messages sent belong to the informational category.

Copyright © 2025 Ping Identity Corporation

About this task

To configure the syslog server:

Steps

1. Configure the IP address or hostname and port number of the syslog server in the `ase.conf` file to send syslog messages to the external server.
2. To stop generating syslog messages, remove the syslog server definition from the `ase.conf` file, and then stop and start ASE.

   Result:

   The following is a snippet from the `ase.conf` file:

   ```
   ; Syslog server settings. The valid format is host:port. Host can be an FQDN or an IPv4
   address.
   syslog_server=
   ```

3. To show the configured syslog server, run the `list_sys_log_server` command.

   Example:

   ```
   /opt/pingidentity/bin/cli.sh list_syslog_server -u admin -p
   192.168.11.108:514, messages sent: 4, bytes sent: 565
   ```

   Result:

   The following is a sample message sent to the syslog server:

   ```
   Aug 16 06:16:49 myhost ase_audit[11944] origin: cli, resource: add_api, info: config_file_path=/opt/
   pingidentity/ase/api.json, username=admin
   Aug 16 06:16:56 myhost ase_audit[11944] origin: cli, resource: list_api, info: username=admin
   ```

Related links

- ASE access, management, and audit logs
- Changing management log levels
- Purge log files

Email alerts and reports

API Security Enforcer (ASE) sends email notifications as either alerts or reports.

Notification type	Description
Alerts	Event-based

Copyright © 2025 Ping Identity Corporation

Notification type	Description
Reports	Sent at a configured frequency from 1 - 7 days using `email_report`

In a cluster deployment, configure the e-mail on the first ASE node. In case the first ASE node is not available, the ASE node with the next highest up-time takes over the task of sending e-mail alerts and daily reports. For more information on ASE cluster, see Administering an ASE cluster.

```
; Defines report frequency in days [0=no reports, 1=every day, 2=once in two days and max is 7 ; days]
email_report=1
; Specify your email settings
smtp_host=smtp://<smtp-server>
smtp_port=587
; Set this value to true if smtp host support SSL
smtp_ssl=true
; Set this value to true if SSL certificate verification is required
smtp_cert_verification=false
sender_email=
sender_password=
receiver_email=

; Defines threshold for an email alert. For example, if CPU usage is 70%, you will get an
; alert.
cpu_usage=70
memory_usage=70
filesystem_size=70
```

Email alerts

When you configure alerts, they use the following template:

```
Event:  <the type of event>
Value:  <the specific trigger for the event>
When:  <the date and time of the event>
Where:  <the IP address or hostname of the server where the event occurred>
```

The following is an example alert you might receive:

```
Event : high memory usage
Value : 82.19%
When : 2019-May-16 18:30:00 PST
Where : vortex-132
```

Email alerts are sent based on the following event categories:

Copyright © 2025 Ping Identity Corporation

System resource

System resources are polled every 30 minutes to calculate usage. An email alert is sent if the value exceeds the defined threshold.The following system resources are monitored:

CPU

Average CPU usage for a 30 minute interval.

Memory

Memory usage at the 30th minute.

Filesystem

Filesystem usage at the 30th minute.

Configuration

When configuration changes occur, an email alert is sent for these events:

- Adding or removing an API
- Adding or deleting a server
- Nodes of a cluster are `UP` or `DOWN`

Decoy API

When decoy APIs are accessed for the first time, an email alert is sent. The time between consecutive alerts is set using `decoy_alert_interval` in the `ase.conf` file. The default value is 180 minutes.

> **Note**
>
> For more information on decoy APIs, see In-Context decoy APIs.

ASE-ABS log transfer and communication

ASE sends an alert in the following two conditions:

Access Log transfer failure

When ASE is unable to send access log files to API Behavioral Security (ABS) for more than an hour, ASE sends an alert with the names of the log files.

ASE-ABS communication failure

When interruptions occur in ASE-ABS communication, an alert is sent identifying the error type. The email also mentions the current and total counter for the alert. The current counter lists the number of times that failure happened in the last hour. The total counter lists the total number of times that error has occurred since ASE was started.

- ABS seed node resolve
- ABS authentication

Copyright © 2025 Ping Identity Corporation

- ABS config post
- ABS cluster INFO
- ABS service unavailable
- Log upload
- Duplicate log upload
- Log file read
- ABS node queue full
- ABS node capacity low
- ABS attack type fetch

The following alerts are logged in the `controller.log` file when email alerts are disabled (`enable_email=false`) in the `ase.conf` file:

- High CPU use
- High memory use
- High filesystem use
- Adding API to ASE
- Removing API from ASE
- Updating and API
- Adding a backend server
- Removing a backend server
- ASE cluster node available
- ASE cluster node unavailable
- Backend server state changed to UP
- Backend server state changed to DOWN
- Log upload service failure
- Error while uploading file
- Invalid ASE license file
- Expired ASE license file

Email reports

ASE sends reports at a frequency in number of days configured in `ase.conf` file. The report is sent at midnight, 00:00:00 hours based on the local system time.

Copyright © 2025 Ping Identity Corporation

The report contains the following:

- Cluster name and location
- Status information on each cluster node:
  - Operating system, IP address, management port, and cluster port
  - Ports and the number of processes (PIDs)
  - Average CPU memory utilization (average during 30-minute polling intervals)
  - Disk usage and log size
- Information on each API: Name, Protocol, and Server Pool

The following example shows what a weekly or daily email report looks like:

Copyright © 2025 Ping Identity Corporation

```
Date: Sat, 29 Jun 2019 04:01:47 -0800 (PST)
To: receiver@example.com
From: sender@exmple.com
Subject:  API Security Enforcer Daily Reports

Dear DevOps,
Please find the daily report generated by ase2 at 2019-Jun-29 00:01:01 UTC.
============== Cluster Details =================
Cluster Name: pi_cluster
Active Nodes: 2
Inactive nodes: 0
No of APIs: 7
LSM State: disabled
Manual IOC: 0
Automated IOC: 0

================== Node 1 ===================
Host Name: apx1
Management Port: 8010
Cluster Port: 8020
Status: Active
Up Since: 2019-Jan-26 09:27:26
Operating System: Ubuntu 14.04.4 LTS
CPU Usage: 55.80%
Memory Usage: 38.17%
Filesystem Usage: 17.20%
Log Size: 20 GB

================== Node 2 ===================
Host Name : apx2
Management Port: 8010
Cluster Port: 8020
Status: Active
Up Since: 2019-Jan-26 09:26:35
Operating System: Ubuntu 14.04.4 LTS
CPU Usage: 55.79%
Memory Usage: 38.17%
Filesystem Usage: 17.20%
Log Size: 20 GB
============================================

================ API Details ==================
API ID: https-app
Status: loaded
Protocol: https
decoy: in-context
Active Servers: 172.17.0.8:2800 172.17.0.7:2700
Inactive Servers:
============================================
API ID: http-app
Status: loaded
Protocol: http
decoy: in-context
Active Servers: 172.17.0.7:2100 172.17.0.8:2300 172.17.0.7:2700
```


Copyright © 2025 Ping Identity Corporation

```
Inactive Servers:
==========================================

Best,
API Security Enforcer
```

Decoy API access reports

ASE sends decoy API access report at a 3-hour interval by default. You can configure this time interval in minutes in the `ase.conf` file by configuring the `decoy_alert_interval` variable. ASE sends the report only if the decoy API is accessed during the configured time interval.

The report provides the following details:

- The start time when the decoy API was first accessed and the end time when it was last accessed
- The ASE cluster name
- The total number of requests for decoy API in the ASE cluster
- The host name of the ASE where the decoy API was accessed

The following example shows what an email report for a decoy API looks like:

```
Date: Sat, 29 Jun 2019 04:01:47 -0800 (PST)
To: receiver@example.com
From: sender@exmple.com
Subject:  API Security Enforcer Decoy Access Reports

Dear DevOps,
Please find the decoy report generated by ase2 at 2019-Jun-29 12:01:45 UTC. The default location for the
decoy log files is in the directory: /opt/pingidentity/ase/logs/
============== Decoy Summary =================
Cluster Name: pi_cluster
Start Time: 2019-Jun-29 09:00:00
End Time: 2019-Jun-29 12:00:00
Total Requests: 875

================== Node 1 ===================
Host Name: ase2
Total Requests: 428

================== Node 1 ===================
Host Name: ase
Total Requests: 447

Best,
API Security Enforcer
```

Copyright © 2025 Ping Identity Corporation

ASE alerts resolution

The following table describes the various email alerts sent by ASE and their possible resolution. The resolution provided is only a starting point to understand the cause of the alert. If ASE is reporting an alert even after the following the resolution provided, contact PingIntelligence for APIs support.

Email alert	Possible cause and resolution
ASE start or restart email	When ASE starts or restarts, it sends an email to the configured email ID. If email from ASE is not received, check the email settings in `ase.conf` file.
High CPU usage	Cause: Each ASE node polls for CPU usage of the system every 30-minutes. If the average CPU usage in the 30-minutes interval is higher than the configured threshold in `ase.conf`, then ASE sends an alert. Resolution: If ASE is reporting a high CPU usage, check if other processes are running on the machine on which ASE is installed. If ASE controller or balancer processes are consuming high CPU, it may mean that ASE is receiving high traffic. You should consider adding more ASE nodes.
High memory usage	Cause: Each ASE node polls for memory usage of the system every 30-minutes. If the average memory usage in the 30-minutes interval is higher than the configured threshold `ase.conf`, then ASE sends an alert. Resolution: If ASE is reporting a high memory usage, check if any other process is consuming memory of the system on which ASE is installed. Kill any unnecessary process other than ASE's process.
High filesystem usage	Cause: Each ASE node polls for filesystem usage of the system every 30-minutes. If the average filesystem usage in the 30-minutes interval is higher than the configured threshold `ase.conf`, then ASE sends an alert. Resolution: If ASE is reporting a high filesystem usage, check if the filesystem is getting full. Run the purge script available in the `util` directory to clear the log files.
API added	ASE sends an email alert when an API is added to ASE using CLI or REST API. Confirm: ASE admin should verify whether correct APIs were added manually or the APIs were added by AAD because of auto-discovery in API Behavioral Security (ABS). If an API is accidentally added, you should immediately remove it from ASE.
API removed	ASE sends an email alert when an API is removed using CLI or REST API. Confirm: ASE admin should verify whether the APIs were deleted intentionally or accidentally.
API updated	ASE sends an email alert when an API definition (the API JSON file) is updated by using CLI or REST API. Confirm: ASE admin should verify whether the correct APIs was updated.
Server added	ASE sends an email alert when a server is added to an API by using CLI or REST API. Confirm: ASE admin should verify whether the correct server was added to API.

Copyright © 2025 Ping Identity Corporation

Email alert	Possible cause and resolution
Server removed	ASE sends an email alert when a server is removed from an API by using CLI or REST API. Confirm: ASE admin should verify whether the correct server was removed from an API.
Cluster node up	ASE sends an email alert when a node joins an ASE cluster. Confirm: ASE admin should verify whether the correct ASE node joined the ASE cluster.
Cluster node down	ASE sends an email alert when a node is removed from an ASE cluster. Confirm: ASE admin should check the reason for removal of ASE node from the cluster. ASE node could disconnect from cluster because of network issues, a manual stop of ASE, or change in IP address of the ASE machine.
Server state changed to Up	ASE sends an email alert when the backend API server changes state from inactive to active. This alert is applicable for Inline ASE when health check is enabled for an API. This is an informative alert.
Server changed to Down	ASE sends an email alert when the backend API server changes state from active to inactive. This alert is applicable for Inline ASE when health check is enabled for an API. Resolution: ASE admin should investigate the reason for the backend API server being not reachable from ASE. You can run the ASE `health_status` command to check the error which caused the server to become inactive.
Decoy API accessed	ASE sends an email alert when a decoy API is accessed. This is an informative alert.

Alerts for uploading access log files to ABS

ASE sends one or more alerts when it is not able to send access log files to ABS. The following table lists the alerts and possible resolution for the alerts.

Email alert	Possible cause and resolution
Network error	Cause: ABS IP may not be reachable or ASE is not able to connect ABS IP and port. Resolution: • If there is a firewall in the deployment, check whether firewall is blocking access to ABS. • Check whether ABS is running. • Check whether correct IP address is provided in the `abs.conf` file.
ABS seed node resolve error	Cause: The host name provided in the `abs.conf` file could not be resolved. Resolution: Check whether correct IP address is provided in `abs.conf` file.
ABS SSL handshake error	Cause: SSL handshake error could be because of an invalid CA certificate. Resolution: Check whether a valid CA certificate is configured in ASE.

Copyright © 2025 Ping Identity Corporation

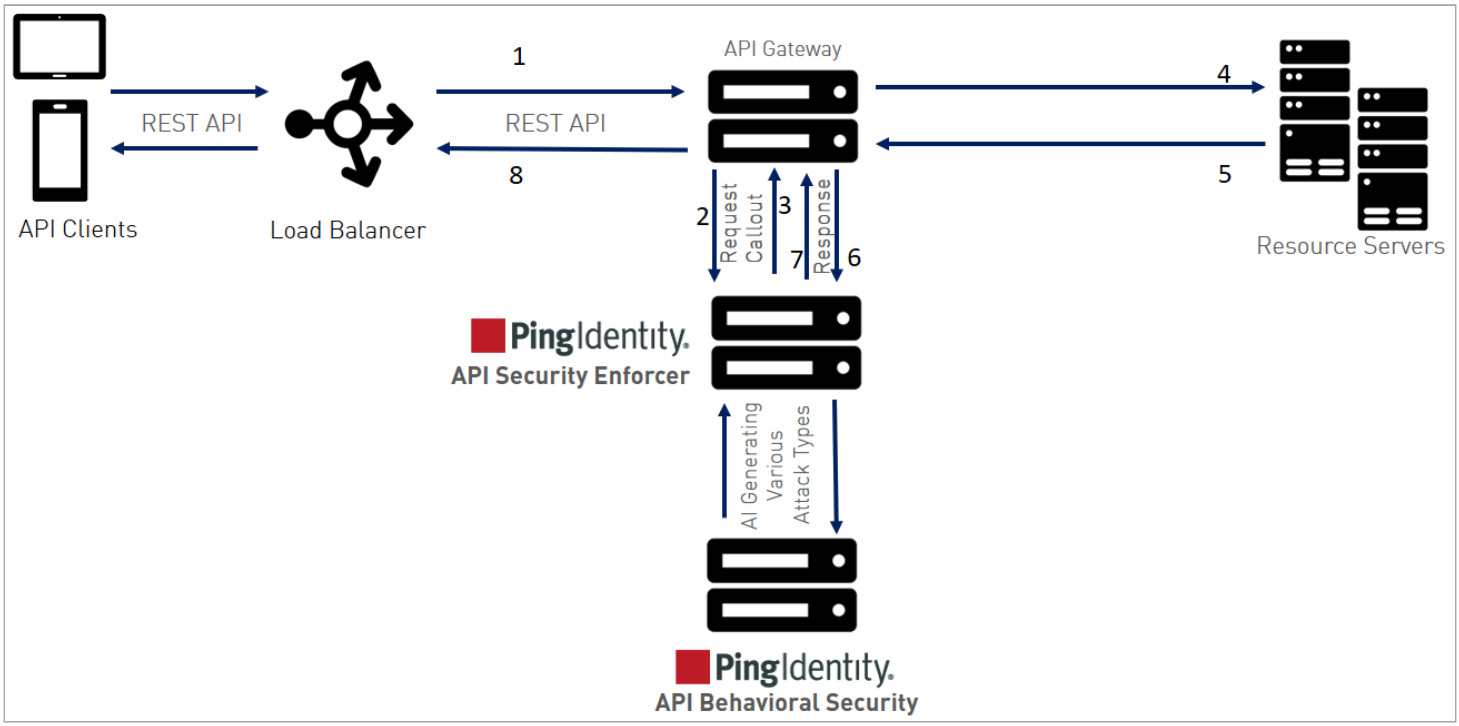
Email alert	Possible cause and resolution
ABS authentication error	Cause: Authentication error could be because of invalid access and secret key. Resolution: Confirm the access key and secret key configured is the same that is configured in ABS `abs.properties` file.
ABS cluster info error	Cause: Error while fetching ABS cluster information. Resolution: Check the `controller.log` file.
ABS config post error	Cause: Error while sending API JSON definition to ABS. Resolution: Check the `controller.log` file.
ABS service unavailable error	Cause: ABS returning `503` response code. Resolution: Check the `abs.log` file.
Log upload error	Cause: API call to upload access log files to ABS fails. Resolution: Check both ASE's `controller.log` and ABS `abs.log` file.
Duplicate log upload error	This is an informative message.
ABS node queue full error	Cause: ABS responds with a message that it's queue is full. This can be because of increased traffic on ASE and large number of access log files being generated. Resolution: Increase the number of ABS nodes.
ABS node capacity low error	Cause: ABS resources are utilized to a maximum. Resolution: Increase the number of ABS nodes.
ABS attack get error	Cause: Error while fetching attack list from ABS. Resolution: Check ASE's `controller.log` file.

Sideband ASE

When deployed in sideband mode, API Security Enforcer (ASE) receives application programming interface (API) calls from an API gateway which passes API traffic information for AI processing. In such a deployment, ASE works along with the API gateway to protect your API environment.

The following diagram and steps describe a typical ASE sideband deployment traffic pattern.

Copyright © 2025 Ping Identity Corporation



1. An incoming request reaches the API gateway.
2. The API gateway makes an API call to send the request metadata in JSON format to ASE.
3. ASE checks the request against a registered set of APIs and checks the origin Internet Protocol (IP) against the AI-generated deny list. If all checks pass, ASE returns a 200-OK response to the API gateway. Otherwise, a different response code is sent to the gateway. The request is also logged by ASE and sent to the AI Engine for processing.
4. If the API gateway receives a 200-OK response from ASE, then it forwards the request to the backend server. If it receives a 403, the gateway does not forward the request to the backend server and returns a different response code to the client.
5. The response from the backend server is received by the API gateway.
6. The API gateway makes a second API call to pass the metadata information to ASE, which sends the information to the API Behavioral Security (ABS) AI engine for processing.
7. ASE receives the metadata information and sends a 200-OK to the API gateway.
8. The API gateway sends the response received from the backend server to the client.

> **Important**
>
> Make sure that XFF is enabled in the API gateway for ASE to detect the client IP addresses correctly.

Configuring ASE sideband mode

About this task

To configure ASE to work in the sideband mode:

Copyright © 2025 Ping Identity Corporation

Steps

1. Edit the `config/ase.conf` file.
2. Set the value of the `mode` parameter to `sideband`.

   The default value of the `mode` parameter is `inline`.

   Example:

   The following is a snippet of the `ase.conf` file with the `mode` parameter set to `sideband`:

   ```
   ; Defines running mode for API Security Enforcer.
   mode=sideband
   ```

Enabling sideband authentication

Enabling sideband authentication ensures a secure the connection between your API gateway and API Security Enforcer (ASE).

About this task

To enable sideband authentication:

Steps

1. Run the `enable_sideband_authentication` command.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh enable_sideband_authentication -u admin -p admin
   Sideband authentication is successfully enabled
   ```

2. To generate an ASE sideband token, run the `create_sideband_token` command.

   Example:

   ```
   /opt/pingidentity/ase/bin/cli.sh create_sideband_token -u admin -p admin
   Sideband token d9b7203c97844434bd1ef9466829e019 created.
   ```

   > **Note**
   >
   > This token is configured in the API gateway for it to communicate securely with ASE.

ase.conf file">

Sideband ASE configuration using the `ase.conf` file

API Security Enforcer (ASE) system-level configuration entails modifying parameters in the `config/ase.conf` file. Some values have default settings that you can modify to support application requirements.

Copyright © 2025 Ping Identity Corporation

The following table provides parameter values and descriptions.

Parameter	Description
ASE mode	
`mode`	Change the mode to `sideband` for ASE to work in a sideband mode. The default value is `inline`.
ASE time zone	
`timezone`	Sets ASE's time zone. The values can be `local` or `UTC`. Default value is `UTC`. If ASE is deployed in a cluster, configure the same time zone on each cluster node manually.
`enable_sideband_keepalive`	When set to `true`, ASE sends a keep-alive in response header for the TCP connection between API gateway and ASE. With the default `false` value, ASE sends a connection close in response header for connection between API gateway and ASE. **Note** This parameter is applicable only when mode is set to `sideband`.
`enable_sideband_authentication`	This parameter only applies in the ASE sideband mode. Set it to `true` to enable authentication with a shared secret between an API gateway and ASE. After setting it to `true`, generate a sideband authentication token using ASE `create_sideband_token` command.
`enable_mtls`	When set to `true`, mutual TLS (MTLS) is enabled for sideband communication between ASE and the Apigee API Gateway. The default is `false`. **Note** This feature requires ASE version 5.1.3 or later.
ASE ports	
`http_ws_port`	Data port used for HTTP or WebSocket protocol. The default value is 8000.
`https_wss_port`	Data port used for HTTPS or Secure WebSocket (wss). The default value is 8443.
`management_port`	Management port used for command-line interface (CLI) and REST API management. The default value is 8010.
ASE administration and audit	
`admin_log_level`	The level of log detail captured. Options include: Fatal – 1, Error – 2, Warning – 3, Info – 4, Debug – 5

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`enable_audit`	When set to `true`, ASE logs all actions performed in ASE in the audit log files. The default value is `true`.
`syslog_server`	Syslog server hostname or IPv4 address:port number. Leave this parameter blank for no syslog generation.
`hostname_refresh`	N/A
`auth_method`	Authentication method used for administrator access. • `ase::db` (Default - Native authentication) • `pam::ldap` (Linux-PAM authentication with script)
`ase_health`	When `true`, enables load balancers to perform a health check using the following URL: `http(s)://<ASE Name>/ase`, where *<ASE Name>* is the ASE domain name The default value is `false`. **Note** Do not configure the `/ase` URL in an API JSON file.
`enable_1G`	N/A
`http_ws_process`	The number of HTTP processes. It is set to `1`. Do not change this value.
`https_wss_process`	The number of HTTPS or processes. It is set to `1`. Do not change this value.
`enable_access_log`	When `true`, log client traffic request and response information. Default value is `true`.
`flush_log_immediate`	When `true`, log files are immediately written to the file system. When `false`, log files are written after a time interval. The default value is `true`.
`attack_list_memory`	The amount of memory used for maintaining allow and deny lists. The default value is 128 MB.
`keystore_password`	Password for the key store. For more information on updating the key store password, see Updating Keystore Password.
`enable_hostname_rewrite`	N/A
ASE cluster	
`enable_cluster`	When `true`, run setup in cluster mode. The default value is `false`, run in standalone mode.
Security	
`enable_sslv3`	When `true`, enable SSLv3. Default value is `false`.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`server_ca_cert_path`	N/A
`enable_xff`	N/A
`enable_firewall`	When `true`, activates the ASE firewall. The default value is `true`.
`enable_strict_request_parser`	When `true`, ASE blocks client http requests with invalid headers start. The default value is `true`.
Real-time API security	
`enable_ase_detected_attack`	When `true`, activates the real-time security in ASE. The default value is `false`.
API deception	
`decoy_alert_interval`	The time interval between decoy API email alerts. The default value is 180 minutes. Maximum value is 1440 minutes (24 hours).
AI-based API Behavioral Security (ABS)	
`enable_abs`	When `true` (default), send access log files to ABS AI Engine for generating API metrics and detecting attacks using machine learning algorithms. Make sure it is set to `true` when ASE is connected to PingOne.
`enable_abs_attack`	When `true` (default), ASE fetches attack list from ABS AI Engine and blocks access by clients in the attack list. When `false`, attack list is not downloaded.
`abs_attack_request_minute`	Time interval in minutes at which ASE fetches ABS attack list. The default value is 10 minutes.
Google Pub/Sub configuration	
`enable_google_pubsub`	Set it to `true` if you want ASE to push metrics data to Google cloud. The default value is `false`. **Note** ASE must be in the `sideband` mode for Google Pub/Sub configuration to take effect.
`google_pubsub_topic`	The path to your topic for publishing and subscribing the messages. For example, `/pingidentity/topic/<your_topic>`, such as `/viatests/topics/ping_incoming`.
`google_pubsub_concurrency`	The number of concurrent connection between ASE and Google Pub/Sub. The maximum value is 1024 connections. Default value is 1000 connections.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`google_pubsub_qps`	The number of messages per second that ASE can publish to the topic. Maximum value is 10,000. The default value is 1000.
`google_pubsub_apikey`	The API Key to establish connection between ASE and Google Pub/Sub. Configuring API Key for Google Pub/Sub is optional.
`cache_queue_size`	The number of messages that are buffered in cache when ASE is not able to publish to Google Pub/Sub. Maximum size of the queue is 10,000 messages. The default value is 300 messages.
`google_pubsub_timeout`	The time in seconds for which ASE tries to publish messages to Google Pub/Sub. In case of failure to publish, ASE makes three attempts to publish the message, after which it writes the message to the `google_pubsub_failed.log` file.
API Publish (ABS)	
`enable_abs_publish`	When `true`, ASE polls ABS to get list of published APIs and list of non-discovered APIs and decide whether APIs received will be added, deleted or updated. When `false`, the published list will not be downloaded. The default value is `false`.
`abs_publish_request_minutes`	This value determines how often ASE will get published API list from ABS. The default value is `10 minutes`.
Alerts and reports	
`enable_email`	When `true`, send email notifications. The default value is `false`. For more information, see Email alerts and reports.
`email_report`	Time interval in days at which ASE sends reports. Minimum value is one day and the maximum is seven days. The default value is `1`.
`smtp_host`	Hostname of SMTP server.
`smtp_port`	Port number of SMTP server.
`smtp_ssl`	Set to `true` if you want email communication to be over SSL. Make sure that the SMTP server supports SSL. If you set `smtp_ssl` to `true` and the SMTP server does not support SSL, email communication falls back to the non-SSL channel. The default value is `true`. Set it to false if email communication is over a non-SSL channel. The email communication will fail if you set the parameter to `false`, but the SMTP server only supports SSL communication.

Copyright © 2025 Ping Identity Corporation

Parameter	Description

`smtp_cert_verification`	**Set to `true` if you want ASE to verify the SMTP server's SSL certificate. The default value is `true`.**
	If you set it to `false`, ASE does not verify SMTP server's SSL certificate; however, the communication is still over SSL.
	Note
	If you have configured an IP address as `smtp_host` and set `smtp_cert_verification` to `true`, then make sure that the certificate configured on the SMTP server has the following:
	`X509v3 extensions:`
	`X509v3 Key Usage:`
	`Key Encipherment, Data Encipherment`
	`X509v3 Extended Key Usage:`
	`TLS Web Server Authentication`
	`X509v3 Subject Alternative Name:`
	`IP Address: X.X.X.X`

`sender_email`	**Email address for sending email alerts and reports.**
`sender_password`	**Password of sender's email account.** **Note** You can leave this field blank if your SMTP server does not require authentication.
`receiver_email`	**Email address to notify about alerts and reports** **See email alerts for more information.**
ASE server resource utilization	
`cpu_usage`	**Percentage threshold value of CPU utilization.** **See email alerts for more information.**
`memory_usage`	**Percentage threshold value of memory usage.** **email alerts alerts for more information.**
`filesystem_size`	**Percentage threshold value of filesystem capacity.** **See email alerts for more information.**
`buffer_size`	**Customizable payload buffer size to reduce the number of iterations required for reading and writing payloads.** **Default value is 16KB. Minimum is 1KB and maximum is 32KB.**

Example

The following is a sample `ase.conf` file:

Copyright © 2025 Ping Identity Corporation

```
; This is API Security Enforcer's main configuration file. This file is in the standard .ini format.
; It contains ports, firewall, log, ABS flags. The comments start with a semicolon (;).

; Defines running mode for API Security Enforcer (Allowed values are inline or sideband).
mode=inline

; Defines http(s)/websocket(s) ports for API Security Enforcer. Linux user should have the privilege to
bind to these ports.
; If you comment out a port, then that protocol is disabled.
http_ws_port=8000
https_wss_port=8443

; REST API
management_port=8010

; For controller.log and balancer.log only
; 1-5 (FATAL, ERROR, WARNING, INFO, DEBUG)
admin_log_level=4

; Defines the number of processes for a protocol.
; The maximum number of allowed process for each protocol is 6 (1 master + 5 child). The
; following defines 1 process for both http/ws and https/wss protocol.
http_ws_process=1
https_wss_process=1

; Enable or disable access logs to the filesystem (request/response).
; WARNING! It must be set to true for sending logs to ABS for analytics.
enable_access_log=true
; To write access log immediately to the filesystem, set to true.
flush_log_immediate=true

; Setting this value to true will enable this node to participate in an API Security Enforcer
; cluster. Define cluster configurations in the cluster.conf
enable_cluster=false

; Current API Security Enforcer version has 3 firewall features: API Mapping, API Pattern
; Enforcement, and Attack Types.
enable_firewall=true

; X-Forwarded For
enable_xff=false

; SSLv3
enable_sslv3=false

; enable Nagle's algorithm (if NIC card is 1G).
enable_1G=true

; tcp send buffer size in bytes(kernel)
tcp_send_buffer_size=65535
; tcp receive buffer size in bytes(kernel)
tcp_receive_buffer_size=65535

; buffer size for send and receive in KBs (user)
```


Copyright © 2025 Ping Identity Corporation

```
buffer_size=16KB

; Set this value to true, to allow API Security Enforcer to send logs to ABS. This
; configuration depends on the value of the enable_access_log parameter.
enable_abs=true

; Set this value to true, to allow API Security Enforcer to fetch attack list from ABS.
enable_abs_attack=true

; This value determines how often API Security Enforcer will get attack list from ABS.
abs_attack_request_minutes=10

; Set this value to true, to allow API Security Enforcer to fetch published API list from ABS.
enable_abs_publish=false

; This value determines how often API Security Enforcer will get published API list from ABS.
abs_publish_request_minutes=10

; Set this value to true, to allow API Security Enforcer to block auto detected attacks.
enable_ase_detected_attack=false

; Set this value to true to enable email for both alerts and daily reports.
enable_email=false

; Defines report frequency in days [0=no reports, 1=every day, 2=once in two days and max is 7 ; days]
email_report=1
; Specify your email settings
smtp_host=smtp://<smtp-server>
smtp_port=587
; Set this value to true if smtp host support SSL
smtp_ssl=true
; Set this value to true if SSL certificate verification is required
smtp_cert_verification=false
sender_email=
sender_password=
receiver_email=

; Defines threshold for an email alert. For example, if CPU usage is 70%, you will get an
; alert.
cpu_usage=70
memory_usage=70
filesystem_size=70

; Authentication method. Format is <auth_agent>::<auth_service>
; Valid values for auth_agent are ase and pam
; ase agent only supports db auth_service
; pam agent can support user configured pam services
; For example ase::db, pam::passwd, pam::ldap etc
auth_method=ase::db

; Enable auditing. Valid values are true or false.
enable_audit=true

; Decoy alert interval in minutes. [min=15, default=3*60, max=24*60]
decoy_alert_interval=180
```

Copyright © 2025 Ping Identity Corporation

```
; Interval for a hostname lookup (in seconds). [min=10, default=60, max=86400]
hostname_refresh=60

; Syslog server settings. The valid format is host:port. Host can be an FQDN or an IPv4
; address.
syslog_server=

; Attack List size in MB or GB. [min=64MB, max=1024GB]
; ASE will take 3*(configured memory) internally. Make sure that the system has at least
; 3*(configured memory) available
; If you are running ASE inside a container, configure the container to use 3*(configured
; memory) shared memory.
attack_list_memory=128MB

; Enable or Disable health check module. ASE uses '/ase' url for both http and https. This is
; useful if ASE is deployed behind a load balancer.
enable_ase_health=false

; Location for server's trusted CA certificates. If empty, Server's certificate will not be
; verified.
server_ca_cert_path=

; enable client side authentication. This setting is applicable only in sideband mode. Once enabled
; request will be authenticated using authentication tokens.
enable_sideband_authentication=false

; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false

; keystore password
keystore_password=OBF:AES:sRNp0W7sSi1zrReXeHodKQ:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerMPwvM4

; enable hostname rewrite for inline mode. ASE will rewrite the host header in request
; to the server's hostname
enable_hostname_rewrite=false

; enable strict parsing checks for client requests
; If enabled, ASE will block request with invalid header start
; If disabled, it will allow requests
; default value = true
enable_strict_request_parser=true

; Set the timezone to utc or local. The default timezone is utc.
timezone=utc

; Google Pub Sub Configuation
enable_google_pubsub=false

google_pubsub_topic=/topic/apimetrics

; Number of concurrent connections to Google Pub/Sub
```


Copyright © 2025 Ping Identity Corporation

```
; Minimum: 1, Default: 1000, Maximum: 1024
google_pubsub_concurrency=1000

; Number of messages published per second.
; Minimum: 1, Default: 1000, Maximum: 10000
google_pubsub_qps=1000

; Google service account API key (Optional)
google_pubsub_apikey=

; Maximum number of messages buffered in memory
; If queue is full, messages are written to logs/google_pubsub_failed.log
; Minimum: 1, Default: 300, Maximum: 10000
cache_queue_size=300

; Timeout in seconds to publish a message to Google Pub/Sub.
; Minimum: 10, Default: 30, Maximum: 300
google_pubsub_timeout=30
```

API naming guidelines

The application programming interface (API) name must follow the following guidelines.

- The name should not have the word “model”.
- The name should not have the word “threshold”.
- The name should not have the word “all”.
- The name should not have the word “decoyall”.

The following is the list of allowed characters in API name:

- The maximum characters in API name can be 160
- - (hyphen), _ (underscore), and white space are allowed in the name
- a-z, A-Z, and 0-9
- The first character must be alphanumeric

Defining an API using API JSON configuration file in sideband mode

To secure your application programming interface (API) environment using sideband API Security Enforcer (ASE) deployment, APIs need to be configured in ASE using an API JavaScript Object Notation (JSON) file.

Each API has a unique API JSON file. ASE ships with sample JSON files located in the `/config/api` directory. You can manually configure the JSON file with the required parameters as shown in the next section.

The API JSON file parameters define the behavior and properties of your API. The sample API JSON files shipped with ASE can be changed to your environment settings and are populated with default values.

The following table describes the JSON file parameters.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`protocol`	API request type with supported values of: `http` - HTTP
`url`	The value of the URL for the managed API. You can configure up to 10 levels of sub-paths when ASE is deployed in sideband mode. For example, `"/shopping"-` name of a 1 level API `"/shopping/electronics/phones/brand" –` 4 level API `"/" –` entire server (used for API Behavioral Security API Discovery or load balancing)
`hostname`	Hostname for the API. The value cannot be empty. `“*”` matches any hostname.
Configure the client identifiers (for example, cookie, API key, OAuth2 token) used by the API	
`cookie`	Name of cookie used by the backend servers.

`cookie_idle_timeout`
`logout_api_enabled`
`cookie_persistence_enabled`
	N/A

`oauth2_access_token`	When `true`, ASE captures OAuth2 Access Tokens. When `false`, ASE does not look for OAuth2 Tokens. Default value is `false`. For more information, see Capture client identifiers in sideband mode.
`is_token_mandatory`	When set to `true`, if the request has a missing token, ASE adds the IP address of the client to deny list and blocks the request. When set to `false`, ASE does not block the client. **Note** For ASE to check and block the client the following values must be set to `true`: • `oauth2_access_token` • `enable_firewall` and `enable_ase_detected_attack` in Sideband ASE configuration using the `ase.conf` file The default value is `false`.
`apikey_qs`	When API key is sent in the query string, ASE uses the specified parameter name to capture the API key value. For more information, see Capture client identifiers in sideband mode.
`apikey_header`	When API key is part of the header field, ASE uses the specified parameter name to capture the API key value. For more information, see Capture client identifiers in sideband mode.
`login_url`	Public URL used by a client to connect to the application.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`enable_blocking`	When `true`, ASE blocks all types of attack on this API. When `false`, no attacks are blocked. Default value is `false`.
`api_mapping`	N/A

API pattern enforcement
`protocol_allowed`
`http_redirect`
`methods_allowed`
`content_type_allowed`
`error_code`
`error_type`
`error_message_body`
	N/A

Flow control
`client_spike_threshold`
`client_connection_queuing`
	N/A

`api_memory_size`	Maximum ASE memory allocation for an API. The default value is 128 MB. The data unit can be MB or GB.

Health_check
`health_check_interval`
`health_retry_count`
`health_url`
	N/A

`server_ssl`	N/A

Servers:
`host`
`port`
	The IP address or hostname and port number of each backend server running the API.

`server_spike_threshold` `server_connection_quota`	N/A

Decoy Config
`decoy_enabled`
`response_code`
`response_def`
`response_message`
`decoy_subpaths`
	When `decoy_enabled` is set to `true`, decoy sub-paths function as decoy APIs .
	`response_code` is the status code (for example `200`) that ASE returns when a decoy API path is accessed.
	`response_def` is the response definition (for example `OK`) that ASE returns when a decoy API path is accessed.
	`response_message` is the response message (for example `OK)` that ASE returns when a decoy API path is accessed.
	`decoy_subpaths` is the list of decoy API sub-paths (for example `shop/admin, shop/root)`.
	See Configuring API deception for details.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`username_header`	**The name of the custom header containing username. When the value of `username_header` is set, ASE extracts the username from the custom header. For more information, see Extract username from custom header in sideband mode.** **Note** You can configure Username capture from either `username_header` or `JWT` object, but not both.

JWT
`location`
`username`
`clientid`
	When the parameter values of `JWT` object are set, ASE decodes the JWT to extract the user information from the JWT object.
	`location` is the place of occurrence of JWT in an API request. The supported values are:
	• `qs:<key name>`
	• `h:<custom header name>`
	• `h:authorization:bearer`
	• `h:authorization:mac`
	• `h:cookie:<cookie key>`
	
	`username` is the JWT claim to extract the username.
	`clientid` is the JWT claim to extract the client-id.
	For more information, see Extract user information from JWT in sideband mode.
	
	Note
	You can configure username capture from either `JWT` object or `username_header`, but not both.

Here is a sample JSON file for a REST API:

Copyright © 2025 Ping Identity Corporation

```
{
"api_metadata": {
"protocol": "http",
"url": "/rest",
"hostname": "*",
"cookie": "",
"cookie_idle_timeout": "200m",
"logout_api_enabled": false,
"cookie_persistence_enabled": false,
"oauth2_access_token": false,
"is_token_mandatory": false,
"apikey_qs": "",
"apikey_header": "",
"login_url": "",
"enable_blocking": true,
"api_mapping": {
"internal_url": ""
},
"api_pattern_enforcement": {
"protocol_allowed": "",
"http_redirect": {
"response_code": "",
"response_def": "",
"https_url": ""
},
"methods_allowed": [],
"content_type_allowed": "",
"error_code": "401",
"error_def": "Unauthorized",
"error_message_body": "401 Unauthorized"
},
"flow_control": {
"client_spike_threshold": "0/second",
"server_connection_queueing": false
},
"api_memory_size": "128mb",
"health_check": false,
"health_check_interval": 60,
"health_retry_count": 4,
"health_url": "/health",
"health_check_headers": {},
"server_ssl": false,
"servers": [
{
"host": "127.0.0.1",
"port": 8080,
"server_spike_threshold": "0/second",
"server_connection_quota": 0
},
{
"host": "127.0.0.1",
"port": 8081,
"server_spike_threshold": "0/second",
"server_connection_quota": 0
}
],
"decoy_config": {
"decoy_enabled": false,
"response_code": 200,
"response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
            "response_message": "",
            "decoy_subpaths": []
            },
            "username_header": "x-username-header",
            "jwt": {
            "location": "h:authorization:bearer",
            "username": "username",
            "clientid": "client_id"
            }
            }
            }
```

> **Note**
>
> The sample JSON file has an extension of `.example`. If you are customizing the example file, then save the file as a `.json` file.

Manually add API JSON to ASE

After configuring an API JSON file, add it to ASE to activate ASE processing. To add an API, execute the following command-line interface (CLI) command:

```
/opt/pingidentity/ase/bin/cli.sh –u admin -p admin add_api {file_path/api_name}
```

After configuring API JSON files for each API, ASE configuration is complete.

Update a configured API JSON

After activation, an API JSON definition can be updated in real time. Edit the API JSON file located in the `/config/api` directory and make the desired changes. Save the edited API JSON file and execute the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh –u admin -p admin update_api <api_name>
```

For example:

```
/opt/pingidentity/ase/bin/cli.sh –u admin -p admin update_api shop
api shop updated successfully
```

Activating API cybersecurity

The API Security Enforcer (ASE) provides real-time application programming interface (API) cybersecurity using the list of attacks generated by the PingIntelligence for APIs AI Engine.

Before you begin

To activate real-time API cybersecurity, you must enable the ASE firewall.

About this task

Copyright © 2025 Ping Identity Corporation

To enable or disable ASE's API cybersecurity feature:

Steps

- To enable ASE's API cybersecurity:

    1. Run the `enable_firewall` command in the ASE command-line interface (CLI).

        Example:

        ```
        /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
        Firewall is now enabled
        ```

    2. After enabling API Security, run the `status` CLI command to verify cybersecurity is enabled.

        Example:

        ```
        /opt/pingidentity/ase/bin/cli.sh status
        Ping Identity Inc., API Security Enforcer
        status : started
        http/ws : port 80
        https/wss : port 443
        firewall : enabled
        abs : disabled
        abs attack : disabled
        audit : enabled
        ase detected attack : disabled
        attack list memory : configured 128.00 MB, used 25.60 MB, free 102.40 MB
        ```

- To disable ASE's API cybersecurity:

    1. Run the `disable_firewall` CLI command.

        Example:

        ```
        /opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_firewall
        Firewall is now disabled
        ```

    2. Run the `status` CLI command to verify that cybersecurity is disabled.

        Example:

        ```
        /opt/pingidentity/ase/bin/cli.sh status
        Ping Identity Inc., API Security Enforcer
        status : started
        http/ws : port 80
        https/wss : port 443
        firewall : disabled
        abs : disabled
        abs attack : disabled
        audit : enabled
        ase detected attack : disabled
        attack list memory : configured 128.00 MB, used 25.60 MB, free 102.40 MB
        ```

Copyright © 2025 Ping Identity Corporation

Enabling and disabling ASE attack detection

API Security Enforcer (ASE) supports real-time ASE attack detection and blocking for application programming interface (API) Deception. ASE blocks hackers that probe a decoy API and try to access a real business API. For more information, see API Deception Environment.

About this task

To enable and disable ASE attack detection:

Steps

- To enable real-time ASE attack detection, run the `enable_ase_detected_attack` command on the ASE command-line interface (CLI).

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_ase_detected_attack
  ```

  Result:

  ASE detected attack is now enabled.

- To disable real-time ASE detected attacks, run the `disable_ase_detected_attack` command on the ASE CLI.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_ase_detected_attack
  ASE detected attack is now disabled
  ```

  Result:

  Attacks are now deleted from the deny list.

Capture client identifiers in sideband mode

The API Security Engine (ASE) identifies attackers for HTTP(s) protocol using five client identifiers:

- Username
- API keys
- OAuth2 token
- Cookie
- IP address

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> ASE supports the extraction of usernames coming in a JSON Web Token (JWT)s or custom headers. For more information, see Extract user information from JWT in sideband mode and Extract username from custom header in sideband mode. ASE can also receive usernames from a gateway policy, when ASE is deployed in a sideband mode. The PingIntelligence for APIs API Behavioral Security (ABS) AI engine identifies them based on metadata logged in ASE's access log files.

The following sections describe how to configure ASE to capture OAuth2 Tokens and API keys.

Configure ASE support for OAuth2 tokens

ASE supports capturing and blocking of OAuth2 tokens. To enable OAuth2 token capture, set the value of `oauth2_access_token` to `true` in the API JavaScript Object Notation (JSON) file. Here is a snippet of an API JSON file with OAuth2 token capture activated. To disable, change the value to `false`.

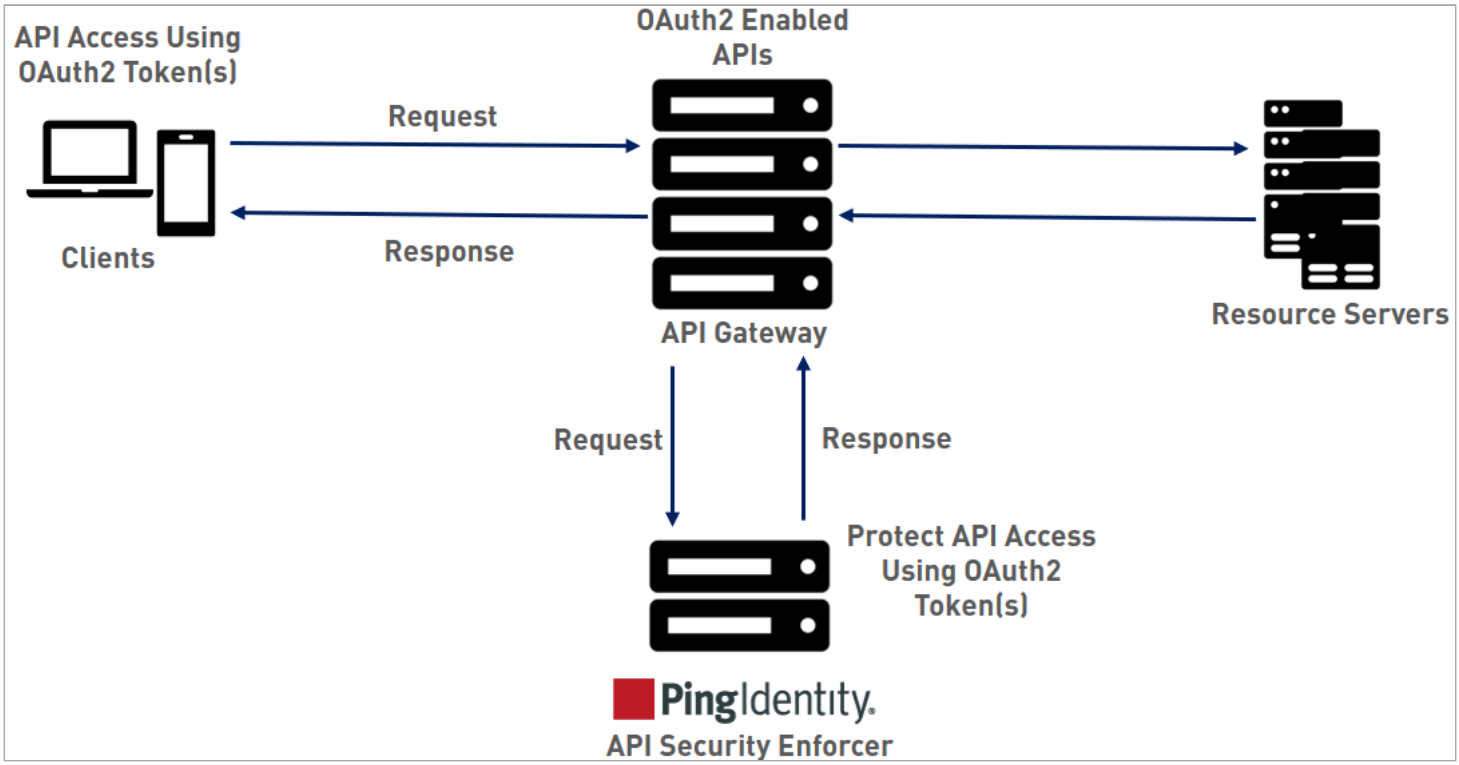
```
"api_metadata": {
        "protocol": "http",
        "url": "/",
        "hostname": "*",
        "cookie": "",
                        "cookie_idle_timeout": "200m",
                        "logout_api_enabled": false,
                        "cookie_persistence_enabled": true,
                        "oauth2_access_token": true,
                        "is_token_mandatory": false,
                        "apikey_qs": "",
                        "apikey_header": "",
                        "login_url": "",
                        "enable_blocking": true,
                        "api_mapping": {
                        "internal_url": ""
                        },
```

When `enable_blocking` is `true`, ASE checks the token against the list of tokens in the allow list and deny list. If the token is in the deny list, the client using the token is immediately blocked. Further, when `is_token_mandatory` is set to `true`, and the incoming request has a missing token, ASE adds the IP address of the client to deny list and blocks the request.

> **Important**
>
> For ASE to check and block the client, `enable_firewall` and `enable_ase_detected_attack` must be set to `true` in Sideband ASE configuration using the `ase.conf` file.

The following diagram shows the traffic flow in an OAuth2 environment:

Copyright © 2025 Ping Identity Corporation



Configure ASE support for API keys

ASE supports the capturing and blocking of API keys. Depending on the API setup, the API key can be captured from the query string or API header. Each API JSON file can be configured with either the query string (`apikey_qs`) or API header (`apikey_header`) parameter.

Here is a snippet of an API JSON file showing an API key being configured to capture the API key from the Query String (`apikey_qs`).

Copyright © 2025 Ping Identity Corporation

```
"api_metadata": {
                                "protocol": "http",
                                "url": "/",
                                "hostname": "*",
                                "cookie": "",
                                "cookie_idle_timeout": "200m",
                                "logout_api_enabled": false,
                                "cookie_persistence_enabled": true,
                                "oauth2_access_token": true,
                                "is_token_mandatory": false,
                                "apikey_qs": "key_1.4",
                                "apikey_header": "",
                                "login_url": "",
                                "enable_blocking": true,
                                "api_mapping": {
                                "internal_url": ""
                                },
```

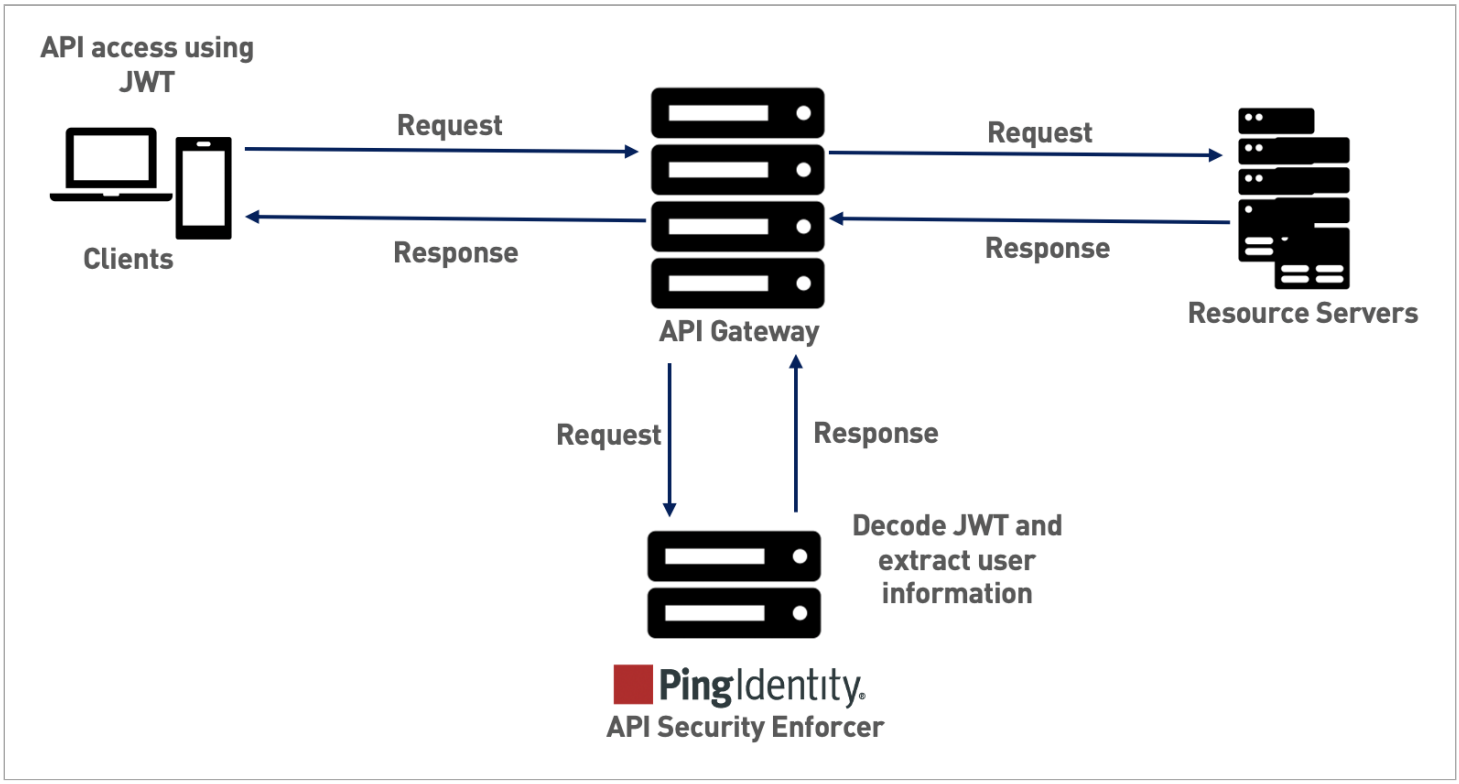
When an API key is included in the API JSON file, ASE supports the blocking of API keys that are manually added to the deny list.

Extract user information from JWT in sideband mode

The API Security Engine (ASE) supports the decoding of transparent JSON Web Token (JWT) received as part of application programming interface (API) requests. ASE extracts the user information from the JWT and logs it in ASE access logs. The API Behavioral Security (ABS) AI Engine analyzes these access logs to generate reports and detect attacks.

The following diagram shows the traffic flow when ASE is in sideband mode.

Copyright © 2025 Ping Identity Corporation



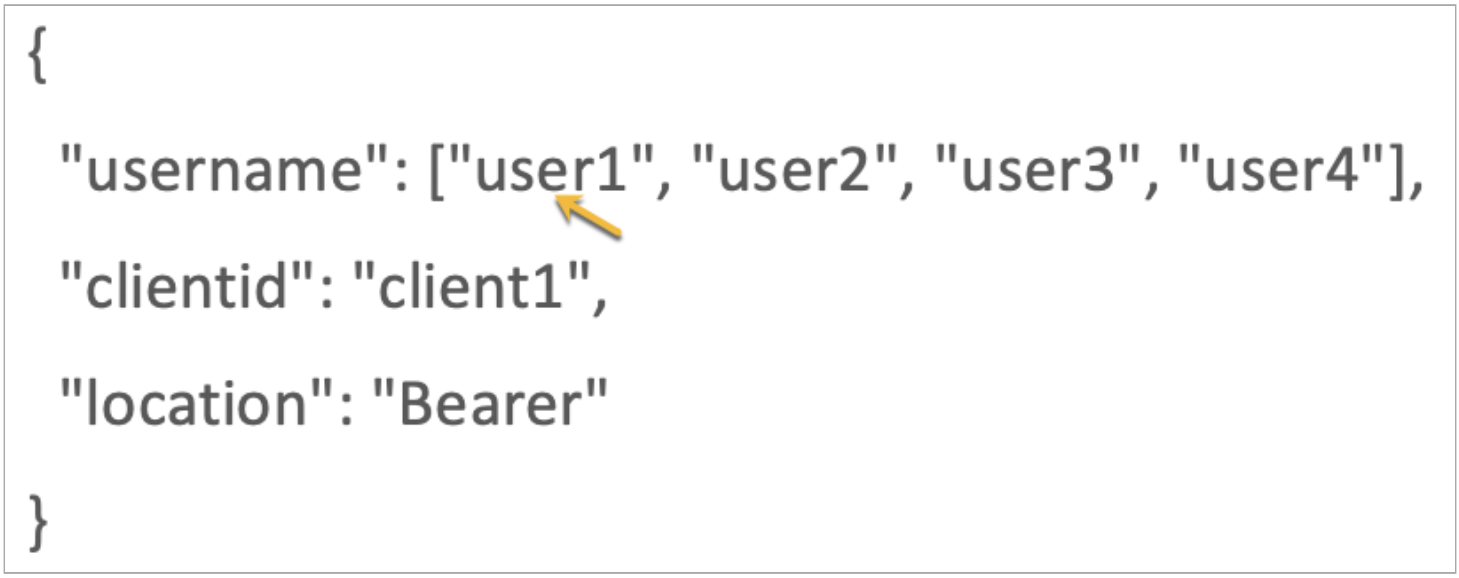
A JWT consists of a header, a payload, and a signature. They are concatenated with periods(.). The following is a sample JWT structure.

ASE decodes the payload to extract user information from a JWT. It can decode JWTs received as part of request headers or query strings. In sideband mode, ASE supports only Bearer scheme in the Authorization header.

> **Note**
>
> ASE does not validate JWTs. It just decodes the JWTs and extracts the user information.

ASE supports a list of usernames in JWT. When the username claim in the payload is an array with multiple elements, ASE extracts the first element of the array. The elements in the array can be strings or numbers and the array should be a valid JavaScript Object Notation (JSON) array.

Copyright © 2025 Ping Identity Corporation



> **Note**
>
> ASE supports arrays only for username claims in the payload. It does not support arrays in clientid or location claims.

When deployed in sideband mode, ASE receives the API request information from the gateway policy and extracts the metadata. The user_info object contains the user information along with other metadata. The following is an example snippet of information received by ASE from API gateway:

```
{
 “source_ip”: “127.0.0.1 ”,
 “source_port”: 12345,
 “method”: “GET”,
 “url”: “/api3?query=eyJ0eXAiOiJKV1QiLCJhbGciHuDXOyfQqAnoXC4bA&abc=xyz”,
 “http_version”: “1.1”,
 “user_info”:[{“username”:“abc”,“client_id”:“cabfsghhbsag”}],
  “headers”: [ { “host”: “shop.com” },
             { “content-type”: “application/xml” },
               { “content-length”: “100” },
               { “x-forwarded-for”: “dev.pxy.com” },
               { “user-agent”: “Mozilla/5.0 (X11; Linux x86_64)
                           AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110 Safari/537.36” }
             ]
}
```

ASE extracts the user information from the user_info object, JWT or both. The following scenarios explain the different ways in which ASE extracts user information:

- **If the gateway policy sends the user_info object with `username` and `clientid`, ASE does not decode the JWT. It extracts the user information from the user_info object.**
- **If the gateway policy sends the user_info object without `username` and `clientid`, ASE decodes the JWT to extract the information.**
- **If the gateway policy sends the user_info object without a `username`, but with `clientid`, ASE decodes the JWT and extracts `username` from the JWT and client identifier from the user_info object.**

Copyright © 2025 Ping Identity Corporation

- If the gateway policy sends the user_info object with a `username`, but without a `clientid`, ASE decodes the JWT to extract `clientid` and captures the `username` from the user_info object.
- If the gateway policy does not send user_info object or sends an invalid user_info object, ASE decodes the JWT to extract the `username` and `clientid` information if available.

> **Note**
>
> If the JWT decoding fails, the API request is not blocked. ASE logs the information got from the gateway policy in the access logs.

The API JSON file

The behavior and properties of your API are defined in an API JSON file in ASE. To enable username capture, set the values for the parameters defined in the JWT object of the API JSON file as per your API setup. For more information, see Defining an API using API JSON configuration file in sideband mode.

The following is an example snippet of an API JSON file:

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/rest",
    "hostname": "*",
    "cookie": "",
    "cookie_idle_timeout": "200m",
    "logout_api_enabled": false,
    "cookie_persistence_enabled": false,
    "oauth2_access_token": true,
    "apikey_qs": "",
    "apikey_header": "",
    "login_url": "",
    "enable_blocking": true,
    "api_mapping": {
      "internal_url": ""
    },
    "username_header": "",

    "jwt": {
      "location": "h:authorization:bearer",
      "username": "username",
      "clientid": "client_id"
    }
  }
}
```

> **Note**
>
> The values assigned to `username` and `clientid` cannot be same.

The following table explains the parameters in the JWT object of API JSON file.

Copyright © 2025 Ping Identity Corporation

Parameter	Description

`location`	`location` is the place of occurrence of JWT in an API request. Configure the parameter with a value applicable to your API.
	The supported values for `location` parameter are:
	
	`qs:<key name>`
	Set the location parameter with this value when JWT occurs as part of a query string and substitute the *<key name>* with the query string parameter. For example, `"location": "qs: access_token"`.
	
	`https://server.example.com/resource?access_token=mF_9.B5f-4.1JqM&p=q`
	
	`h:<custom header name>`
	Set the location parameter with this value when JWT is part of a custom header and substitute the *<custom header name>* with custom header. For example, `"location": "h:X-jwt-header"`.
	
	`X-jwt-header: eyJhbGcUzI1NiI.eyJzDkwIG4gRG9xpZWQiOjwMjJ9.DWw5PDZEl-g`
	
	`h:Authorization:bearer`
	Set the location parameter with this value when JWT is part of Authorization header, with bearer scheme. For example, `"location": "h:Authorization:bearer"`.
	
	`Authorization: Bearer eyJhbGIUzIiI.eyJzdiIxG4gRG9lIiwiZiOjJ9.DWPwNDZEl-g`
	
	`h:cookie:<cookie key>`
	Set the location parameter with this value when JWT occurs as part of a cookie and substitute the *<cookie key>* with the cookie name. For example, `"location": "h:cookie: access_token"`.
	
	`Cookie: access_token=eyJhbGiIsI.eyJpc3MiOiJodHRwczotcGxlL.mFrs3ZodqKP4F1cB`

`username`	The JWT claim to extract the username.
`clientid`	The JWT claim to extract the client identifier.

When `enable_blocking` is set to `true`, ASE checks the username against the list of usernames in the allow list and deny list. If the username is in the deny list, the client using the username is blocked.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> ASE also supports extracting a username from a custom HTTP header. However, you can configure username capture from either custom header or JWT, but not both. For more information, see Extract username from custom header in sideband mode.

The API discovery process

The ABS AI Engine processes the ASE access logs and discovers new and unknown APIs in your environment. A root API JSON is defined in ASE to enable API discovery by ABS. If the root API JSON has a JWT object configured with values set for all the keys, then the APIs discovered by the ABS will have the JWT object. For more information on API discovery, see API discovery and configuration.

The following table explains the behavior of ASE when the root API JSON has an incomplete JWT object. It also describes its impact on the APIs discovered by ABS in your environment.

Scenarios	Behavior of ASE	API discovery
When a JWT object is not configured in `root` API JSON.	ASE processes the `root` API JSON file.	A JWT object gets added to the discovered APIs with all the keys but empty values. For example: `"jwt": { "username": "", "clientid": "", "location": "" }`
When a JWT object is configured in the `root` API JSON file, but with no keys. For example: `“jwt”:{}`	ASE does not process the `root` API JSON file.	The API is not discovered.
When a JWT object is configured with all the keys present but no values set. For example: `"jwt": { "username": "", "clientid": "", "location": "" }`	ASE processes the `root` API JSON file.	A JWT object gets added to the discovered APIs with all the keys but empty values. For example: `"jwt": { "username": "", "clientid": "", "location": "" }`

Copyright © 2025 Ping Identity Corporation

Scenarios	Behavior of ASE	API discovery
When a JWT object is configured but not all keys are set. For example: `"jwt": { "username": "", "location": "" }`	ASE does not process the `root` API JSON file.	The API is not discovered.

> **Note**
>
> The API JSON file shipped with ASE is compatible with earlier versions of API JSON files. ASE automatically adds an empty JWT object to the API JSON file to maintain compatibility.

Extract username from custom header in sideband mode

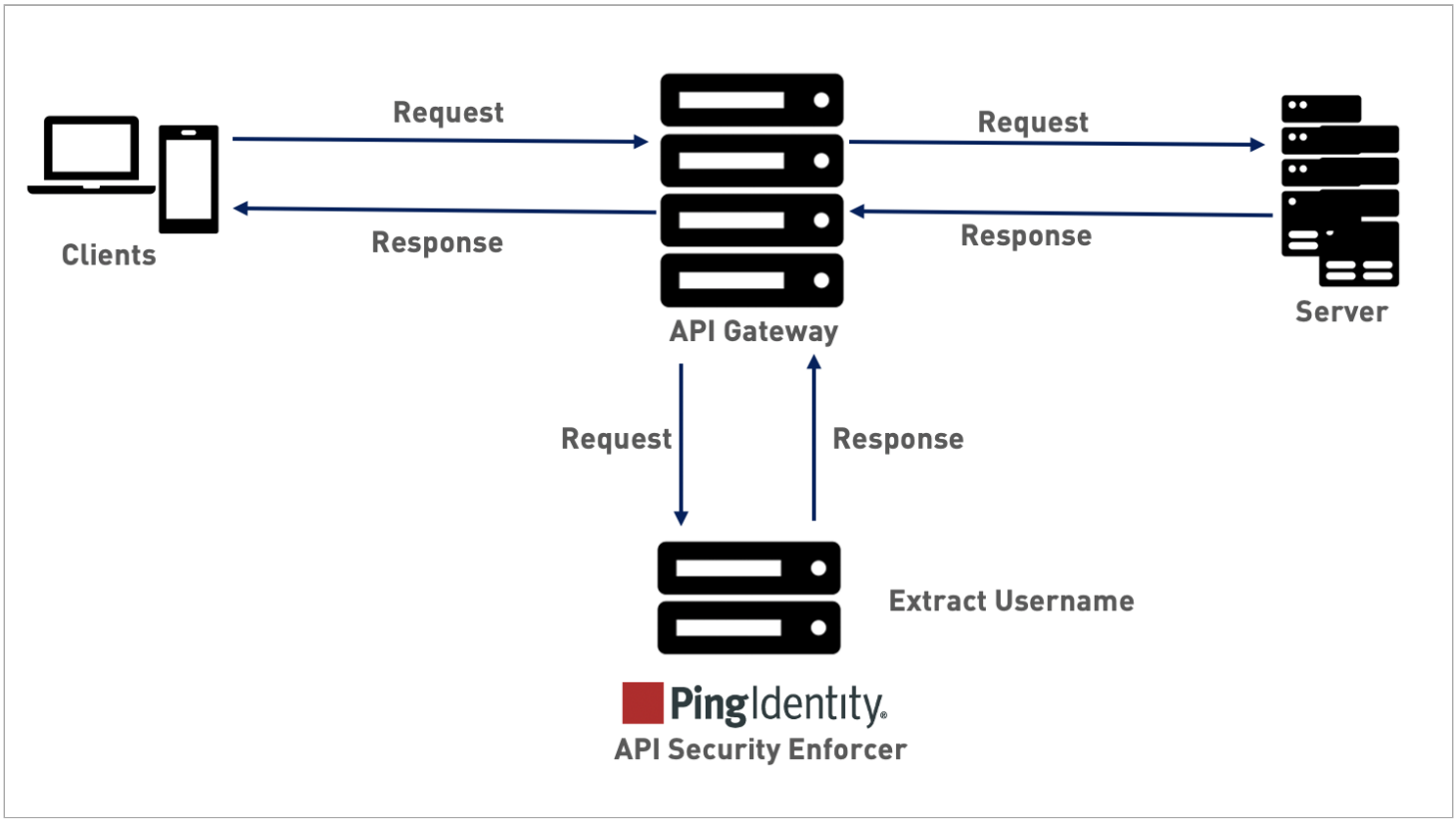
This topic discusses the extraction of username from a custom header when API Security Enforcer (ASE) is in sideband mode. ASE supports capturing usernames from custom headers in a request. It extracts the username and logs it in ASE access logs. ASE sends these access log files to the API Behavioral Security (ABS) AI Engine to detect attacks.

Following is an example snippet of username information logged to ASE access log:

```
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [connectinfo:127.0.0.0:80]
[type:connection] connection received
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:request] [api_id:api1] GET /abcd HTTP/1.1
x-username-header: 12n4uf9ckls

host: http://pi-api-mngmnt.azr-api.net/
accept: /
content-type: text/plain;charset=UTF-8

[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:backend_info] [backend_type:nonssl] [0] [api_id:api1] [hostname:not available] backend
selected
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:req_payload] [api_id:api1] [size:0]
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:user_info] [api_id:api1] username: 12n4uf9ckls
```

When deployed in sideband mode, ASE receives the application programming interface (API) request information from the sideband policy and extracts the metadata like user information, Internet Protocol (IP) addresses and so on. The following diagram shows the traffic flow when ASE is in sideband mode.

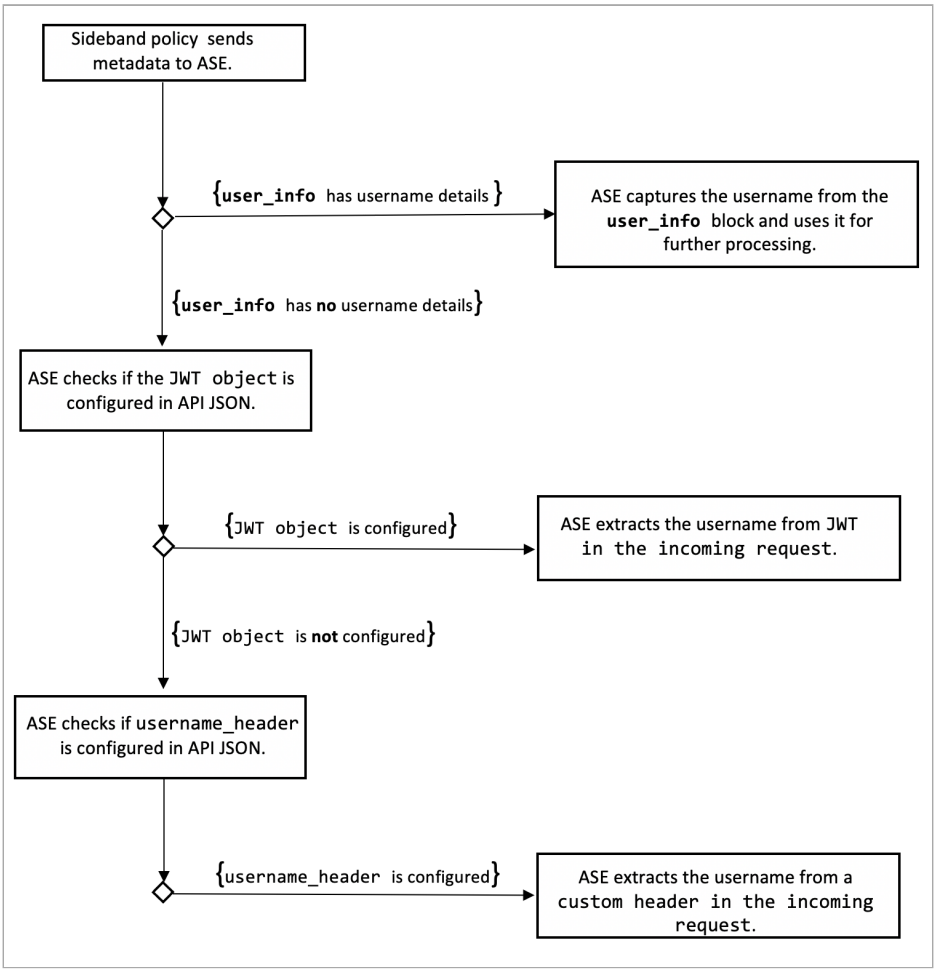
Copyright © 2025 Ping Identity Corporation



The sideband policy sends user information in a `user_info` object to ASE. If the `user_info` object contains username, then ASE extracts it. Otherwise, ASE checks the API JSON configuration.

The API JSON can be configured to extract username from either a JSON Web Token (JWT) or a custom header. ASE first checks the `JWT` object. If it is configured, then ASE extracts the username from the JWT in an incoming request. If the `JWT` object is not configured, then ASE checks the `username_header` parameter configuration in the API JSON file. If it is set, ASE extracts the username from the custom header that comes as part of an incoming request. For more information, see Configure API JSON section.

Copyright © 2025 Ping Identity Corporation



Important

ASE supports extracting username from either JWTs or custom headers. You can configure API JSON to capture username from either custom header or JWT, but not both for a given API. For more information on extracting usernames from JWTs, see Extract user information from JWT in sideband mode.

API JSON configuration in sideband mode

The behavior and properties of your API are defined in an API JSON file in the ASE. To enable username capture from a custom header, set the value of the `username_header` parameter to the custom header name containing the username. The following is an example snippet of an API JSON file.

Copyright © 2025 Ping Identity Corporation

```
{
    "api_metadata": {
        "protocol": "http",
        "url": "/",
        "cookie": "JSESSIONID",
        "hostname": "*",
        "oauth2_access_token": false,
        "apikey_qs": "",
        "apikey_header": "",
        "enable_blocking": true,

        "cookie_idle_timeout": "200m",
        "logout_api_enabled": false,
        "cookie_persistence_enabled": false,
        "login_url": "",
        "api_mapping": {
            "internal_url": ""
        },
        "api_pattern_enforcement": {
            "protocol_allowed": "",
            "http_redirect": {
                "response_code": "",
                "response_def": "",
                "https_url": ""
            },
            "methods_allowed": [],
            "content_type_allowed": "",
            "error_code": "401",
            "error_def": "Unauthorized",
            "error_message_body": "401 Unauthorized"
        },
        "flow_control": {
            "client_spike_threshold": "0/second",
            "server_connection_queueing": false
        },
        "api_memory_size": "128mb",
        "health_check": false,
        "health_check_interval": 60,
        "health_retry_count": 4,
        "health_url": "/",
        "health_check_headers": {},
        "server_ssl": false,
        "servers": [],
        "decoy_config": {
            "decoy_enabled": false,
            "response_code": 200,
            "response_def": "",
            "response_message": "",
            "decoy_subpaths": []
        },
        "username_header": "x-username-header",

        "jwt": {
            "location": "",
            "username": "",
            "clientid": ""
        }
    }
}
```


Copyright © 2025 Ping Identity Corporation

For more information, see Defining an API using API JSON configuration file in sideband mode.

You can optionally block a client. When `enable_blocking` is set to *true*, ASE checks the username against the list of usernames in the allow list and deny list. If the username is in the deny list, the client using the username is blocked.

> **Note**
>
> The API JSON file shipped with ASE is compatible with earlier versions of API JSON files. ASE automatically adds an optional `username_header` parameter to the API JSON file to maintain compatibility.

Managing allow lists and deny lists

The API Security Enforcer (ASE) maintains both allow lists and deny lists.

Allow list

List of safe IP addresses, cookies, OAuth2 tokens, API keys, or usernames that are not blocked by ASE.The list is manually generated by adding the client identifiers using command-line interface (CLI) commands.

Deny list

List of bad IP addresses, cookies, OAuth2 tokens, API keys, or usernames that are always blocked by ASE.The list consists of entries from one or more of the following sources:

- API Behavioral Security (ABS)-detected attacks, such as data exfiltration. ABS-detected attacks have a time-to-live (TTL) in minutes. The TTL is configured in ABS.
- ASE-detected attacks, such as invalid method or decoy API accessed.
- List of bad clients manually generated by CLI.

Copyright © 2025 Ping Identity Corporation

Allow list

Managing the allow list
About this task

To manage operations for OAuth2 Tokens, cookies, IP addresses, API keys, and usernames on an allow list:

Steps

- To add an IP address to an allow list, run the `add_whitelist` command with the `ip` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist ip 10.10.10.10
  ip 10.10.10.10 added to whitelist
  ```

- Add a cookie to an allow list, run the `add_whitelist` command with the `cookie` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist cookie JSESSIONID cookie_1.4
  cookie JSESSIONID cookie_1.4 added to whitelist
  ```

- To add a token to an allow list, run the `add_whitelist` with the `token` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist token token1.4
  token token1.4 added to whitelist
  ```

- To add an API key to an allow list, run the `add_whitelist` command with the `api_key` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist api_key X-API-KEY key_1.4
  api_key X-API-KEY key_1.4 added to whitelist
  ```

- To add a username to an allow list, run the `add_whitelist` command with the `username` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist username abc@example.com
  username abc@example.com added to whitelist
  ```

- To view an allow list, run the `view_whitelist` command.

Copyright © 2025 Ping Identity Corporation

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_whitelist
Whitelist
1) type : ip, value : 1.1.1.1
2) type : cookie, name : JSESSIONID, value : cookie_1.1
3) type : token, value : token1.3
4) type : api_key, name : X-API-KEY, value : key_1.4
5) type : username, value : abc@example.com
```

- To delete an entry from an allow list, run the `delete_whitelist` command.

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist ip 4.4.4.4
ip 4.4.4.4 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist cookie JSESSIONID
cookie_1.1
cookie JSESSIONID cookie_1.1 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist token token1.1
token token1.1 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist api_key X-API-KEY key_1.4
api_key X-API-KEY key_1.4 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist username abc@example.com
```

- To clear the allow list, run the `clear_whitelist` command.

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin clear_whitelist
This will delete all whitelist Attacks, Are you sure (y/n) : y
Whitelist cleared
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin clear_whitelist
This will delete all whitelist Attacks, Are you sure (y/n) : n
Action canceled
```

Copyright © 2025 Ping Identity Corporation

Deny list

Managing the deny list
About this task

To manage IP addresses, Cookies, OAuth2 Tokens, and API keys on a deny list:

Steps

- To add an IP address to the deny list.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist ip 1.1.1.1
  ip 1.1.1.1 added to blacklist
  ```

- To add a cookie to a deny list, run the `add_blacklist` command with the `cookie` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist cookie JSESSIONID
  ad233edqsd1d23redwefew
  cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
  ```

- To add a token to a deny list, run the `add_blacklist` command with the `token` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token ad233edqsd1d23redwefew
  token ad233edqsd1d23redwefew added to blacklist
  ```

- To add an API key to a deny list, run the `add_blacklist` command with the `api_key` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist api_key AccessKey
  b31dfa4678b24aa5a2daa06aba1857d4
  api_key AccessKey b31dfa4678b24aa5a2daa06aba1857d4 added to blacklist
  ```

- To add a username to a deny list, run the `add_black list` command with the `username` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist username abc@example.com
  username abc@example.com added to blacklist
  ```

  You can also add username with space to a deny list. For example, `your name`.

- To view the entire deny list, run the `view_blacklist` command with the `all` option.

Copyright © 2025 Ping Identity Corporation

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist all
Manual Blacklist
1)  type : ip, value : 172.168.11.110
2)  type : token, value : cdE94R3osh283B7NoiJR41XHgt7gxroot
3)  type : username, value : blockeduser
4)  type : cookie, name : JSESSIONID, value : pZlhg5s3i8csImMoas7vh81vz
5)  type : api_key, name : x-api-key, value : d4d28833e2c24be0913f4267f3b91ce5
ABS Generated Blacklist
1)  type : token, value : fAtTzxFJZ2Zkr7HZ9KM17s7kY2Mu
2)  type : token, value : oFQOr11Gj8cCRv1k4849RZOPztPP
3)  type : token, value : Rz7vn5KoLUcAhruQZ4H5cE00s2mG
4)  type : token, value : gxbkGPNuFJw69Z5PF44PoRIfPugA
5)  type : username, value : user1
Realtime Decoy Blacklist
1)  type : ip, value : 172.16.40.15
2)  type : ip, value : 1.2.3.4
```

Note

You can view the entire deny list or based on the type of real-time violation.

- To view the deny list based on decoy IP addresses, run the `view_blacklist` with the `decoy` option.

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist decoy
Realtime Decoy Blacklist
1) type : ip, value : 4.4.4.4
```

- To view the deny list based on protocol violations, run the `view_blacklist` with the `invalid_protocol` option.

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist invalid_protocol
Realtime Protocol Blacklist
1) type : token, value : token1.1
2) type : ip, value : 1.1.1.1
3) type : cookie, name : JSESSIONID, value : cookie_1.1
```

- To view the deny list based on method violations, run the `view_blacklist` with the `invalid_method` option.

Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist invalid_method
Realtime Method Blacklist
1) type : token, value : token1.3
2) type : ip, value : 3.3.3.3
3) type : cookie, name : JSESSIONID, value : cookie_1.3
```

Copyright © 2025 Ping Identity Corporation

- To view the deny list based on content-type violation, run the `view_blacklist` with the `invalid_content_type` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist invalid_content_type
  Realtime Content-Type Blacklist
  1) type : token, value : token1.2
  2) type : ip, value : 2.2.2.2
  3) type : cookie, name : JSESSIONID, value : cookie_1.2
  ```

- To view ABS-detected attacks, run the `view_blacklist` with the `abs_detected` option.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist abs_detected
  No Blacklist
  ```

- To delete an entry from a deny list, run the `delete_blacklist` command.

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_blacklist ip 1.1.1.1
  ip 1.1.1.1 deleted from blacklist
  ./bin/cli.sh -u admin -p admin delete_blacklist cookie JSESSIONID avbry47wdfgd
  cookie JSESSIONID avbry47wdfgd deleted from blacklist
  ./bin/cli.sh -u admin -p admin delete_blacklist token 58fcb0cb97c54afbb88c07a4f2d73c35
  token 58fcb0cb97c54afbb88c07a4f2d73c35 deleted from blacklist
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_blacklist api_key AccessKey
  b31dfa4678b24aa5a2daa06aba1857d4
  ```

- To clear the deny list, run the `clear_blacklist` command.

  > **Warning**
  >
  > When clearing the deny list, make sure that the real-time ASE detected attacks and ABS detected attacks are disabled. If these are not disabled, the deny list gets populated again as both ASE and ABS are continuously detecting attacks.

  Example:

  ```
  ./bin/cli.sh -u admin -p admin clear_blacklist
  This will delete all blacklist Attacks, Are you sure (y/n) :y
  Blacklist cleared
  ./bin/cli.sh -u admin -p admin clear_blacklist
  This will delete all blacklist Attacks, Are you sure (y/n) :n
  Action canceled
  ```

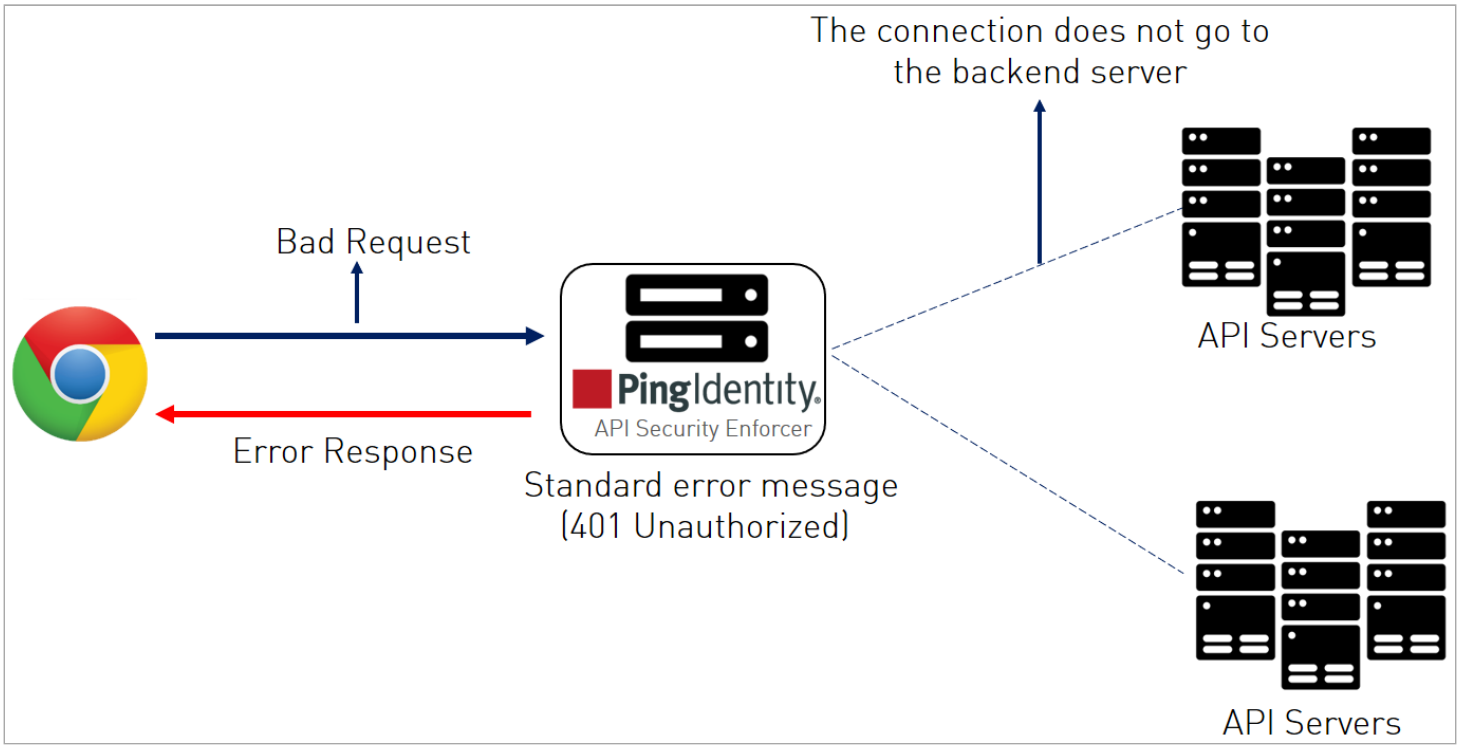

Copyright © 2025 Ping Identity Corporation

ASE generated error messages for blocked requests

The API Security Enforcer (ASE) blocks certain requests based on application programming interface (API) Mapping or API Behavioral Security (ABS)-detected attacks. For these blocked requests, it sends a standard error message back to the client.

The following table describes the error messages.

Blocked Connection	HTTP Error Code	Error Definition	Message Body
Unknown API	503	Service Unavailable	Error: Unknown API
Unknown Hostname	503	Service Unavailable	Error: Unknown Hostname
Malformed Request	400	Bad Request	Error: Malformed Request
IP attack	403	Unauthorized	Error: Unauthorized
Cookie attack	403	Unauthorized	Error: Unauthorized
OAuth2 Token attack	403	Unauthorized	Error: Unauthorized
API Key attack	403	Unauthorized	Error: Unauthorized
Username attack	403	Unauthorized	Error: Unauthorized



Copyright © 2025 Ping Identity Corporation

Enabling per API blocking

You can configure the API Security Enforcer (ASE) to selectively block on a per application programming interface (API) basis by configuring an API JavaScript Object Notation (JSON) file parameter.

About this task

To enable per API blocking for each API:

Steps

- Set the `enable_blocking` parameter to `true` in the API JSON file.

  Example:

```
api_metadata": {
 "protocol": "http",
 "url": "/",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "enable_blocking": true,
 "login_url": "",
 "api_mapping": {
 "internal_url": ""
 },
```

> **Note**
>
> If per API blocking is disabled, API Behavioral Security (ABS) still detects attacks for that specific API. ASE does not block them. ASE continues to block attacks on other APIs with the `enable_blocking` parameter set to `true`.

API deception environment in sideband mode

A decoy application programming interface (API) is configured in API Security Enforcer (ASE) and the API gateway. It requires no changes to backend servers. It appears as part of the API ecosystem and is used to detect the attack patterns of hackers.

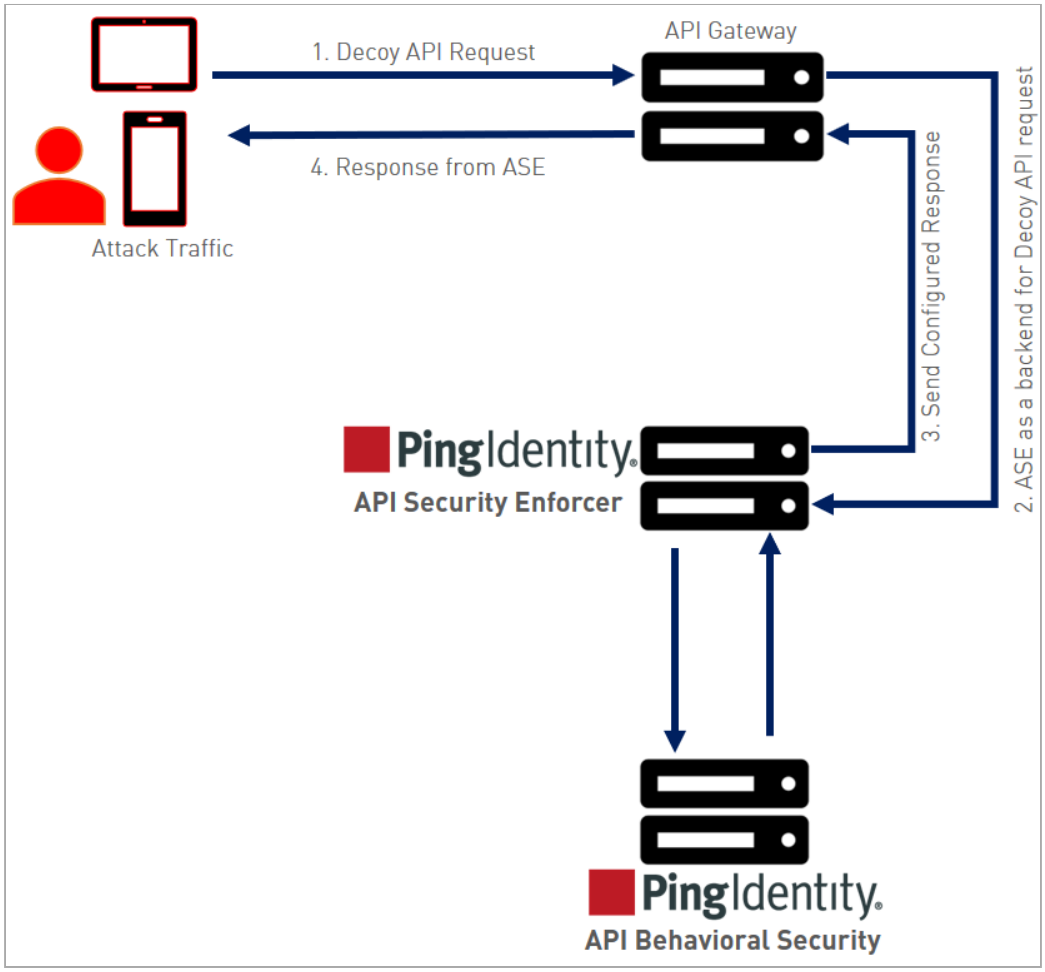
When a hacker accesses a decoy API, ASE sends a predefined response (defined in the `response_message` parameter in the API file) to the client request and collects the request information as a footprint to analyze API ecosystem attacks. ASE acts as a backend for decoy APIs configured in the API gateway.

Decoy API traffic is separately logged in files named with the following format: `decoy_pid_<pid_number>yyyy-dd-mm-<log_file_rotation_time` *(for example:* `decoy_pid_87872017-04-04_10-57.log`*. Decoy log files are rotated every 24 hours and stored in the* `opt/pingidentity/ase/logs` *directory.*

Copyright © 2025 Ping Identity Corporation

Decoy APIs are independent APIs where every path is a decoy API. Any sub- paths accessed in the API are treated as part of the decoy API. The figure shows an example.

> **Important**
>
> *In sideband ASE deployment, you can only configure out-of-context decoy APIs.*



The following steps explain the flow of decoy API traffic:

1. *The attacker sends decoy API request*
2. *API gateway forwards the request is to the configured decoy API which is ASE functioning as a backend server for the decoy API.*
3. *The configured response is sent to the API gateway.*
4. *The configured response from ASE is sent back to the attacker.*

The decoy request is logged in `decoy.log` *file and sent to PingIntelligence API Behavioral Security (ABS) AI Engine for further analysis. The following is a snippet of an API JSON file which has been deployed as an out-of-context decoy API:*

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",
 "url": "/account",
 "hostname": "*",
;
; Note – other configuration parameters removed
;
 "decoy_config":
 {
 "decoy_enabled": true,
 "response_code" : 200,
 "response_def" : "OK",
 "response_message" : "OK", decoy API configuration
 "decoy_subpaths": [

 ]
 }
```

Since the `decoy_subpaths` *parameter is empty, any sub-path accessed by the attacker after* `/account` *is regarded as a decoy path or decoy API.*

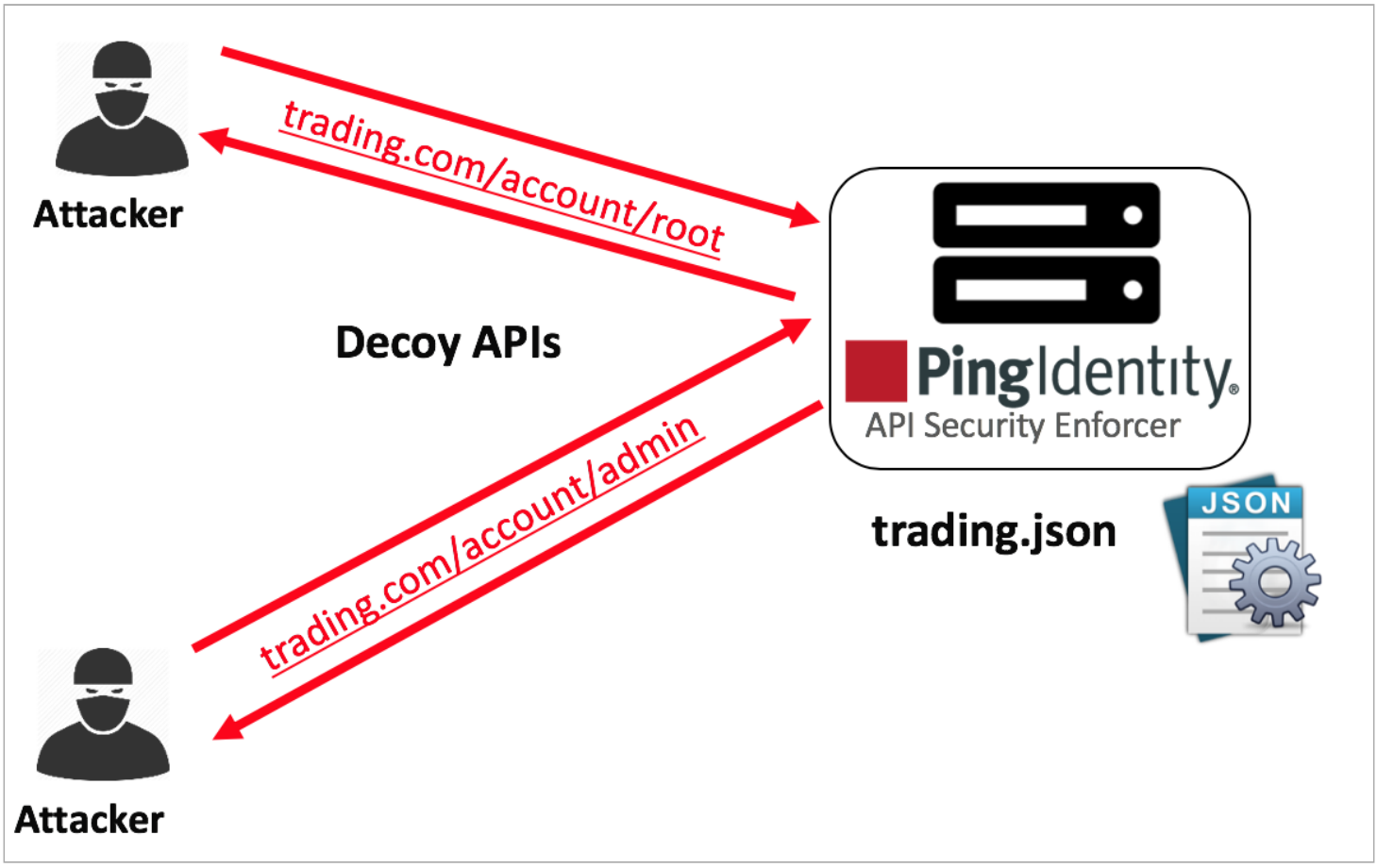
After configuring a decoy API, check the API listings by running the `list_api` *command:*

```
opt/pingidentity/ase/bin/cli.sh list_api -u admin -p
flight ( loaded ), https
trading ( loaded ), https, decoy: out-context
```

Out-of-context decoy API

Out-of-context decoy APIs are independent APIs where every path is a decoy API. Any sub-paths accessed in the API are treated as part of the decoy API.

Copyright © 2025 Ping Identity Corporation



The following is a snippet of a trading API JSON file which has been deployed as a decoy API:

```
{
    "api_metadata": {
        "protocol": "http",
        "url": "/account",
        "hostname": "*",
;
; Note – other configuration parameters removed
;
        "decoy_config":
        {
          "decoy_enabled": true,
          "response_code" : 200,
          "response_def" : "OK",
          "response_message" : "OK",          Decoy API Configuration
          "decoy_subpaths": [

          ]
       }
```

Since the `decoy_subpaths` parameter is empty, any sub-path accessed by the attacker after `/account` is regarded as a decoy path or decoy API.

After configuring in-context or out-of-context decoy APIs, you can check the API listings by running the `list_api` command:

Copyright © 2025 Ping Identity Corporation

```
opt/pingidentity/ase/bin/cli.sh list_api -u admin -p
flight ( loaded ), https
shop ( loaded ), https, decoy: in-context
trading ( loaded ), https, decoy: out-context
```

Real-time API deception attack blocking

API Security Enforcer (ASE) detects any client probing a decoy application programming interface (API). When a client probes an out-of-context decoy API, ASE logs but does not drop the client connection. However, if the same client tries to access a legitimate path in the in-context decoy API, then ASE blocks the client in real-time.

Here is a snippet of an ASE access log file showing real-time decoy blocking:

```
[Tue Aug 14 22:51:49:707 2018] [thread:209] [info] [connectionid:1804289383] [connectinfo:
100.100.1.1:36663] [type:connection_drop] [api:decoy] [request_payload_length:0] GET /decoy/test/test HTTP/
1.1
User-Agent: curl/7.35.0
Accept: /
Host: app
```

The blocked client is added to the deny list which can be viewed by running the `view_blacklist` CLI command:

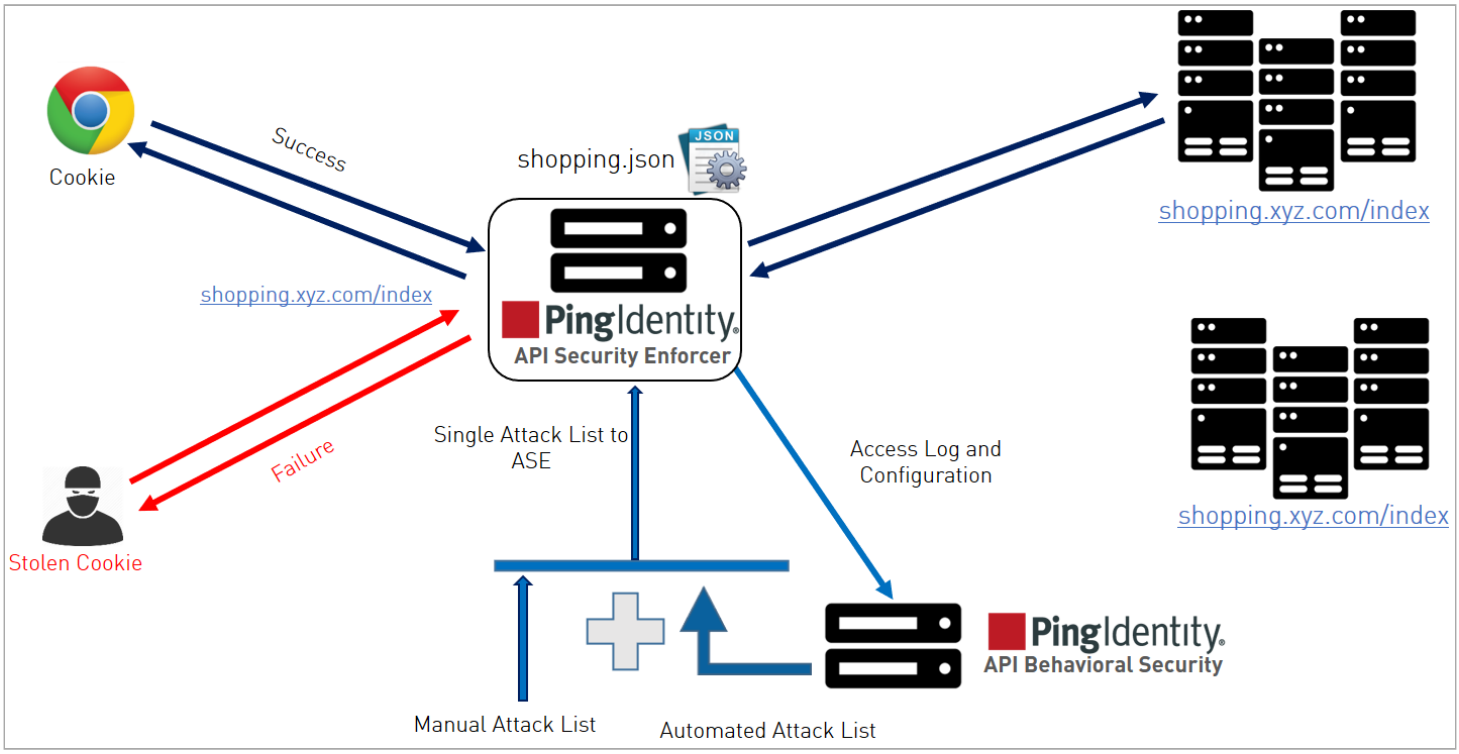
```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist
Realtime Decoy Blacklist
1) type : ip, value : 100.100.1.1
```

ABS AI-based security

The API Behavioral Security (ABS) AI Engine detects attacks using artificial intelligence (AI) algorithms.

After receiving API Security Enforcer (ASE) access logs and application programming interface (API) JavaScript Object Notation (JSON) configuration files, ABS applies AI algorithms to track API connections and detect attacks. If the `enable_abs_attack` parameter is set to `true`, ABS sends the deny list to ASE, which blocks client identifiers, such as API keys, usernames, cookie, Internet Protocol (IP) address, and OAuth token on the list.

Copyright © 2025 Ping Identity Corporation



ASE to ABS connectivity

To connect the API Security Enforcer (ASE) to API Behavioral Security (ABS), configure the ABS address (IPv4:<*port*> or <*hostname*>:<*port*>), access key, and secret key in the `abs.conf` file located in the `/<ASE installation path>/pingidentity/ase/config` directory.

> **Important**
>
> The `enable_abs` parameter must be set to `true` in the `ase.conf` file. When ABS is in a different AWS security group, use a private IP address.

The following table includes the parameter values and descriptions.

Parameter	Description
`deployment_type`	The ABS deployment mode. Valid values are `cloud` or `onprem`. The default value is `onprem`.
`gateway_credential`	This parameter is used when ABS is deployed in `cloud` mode. The credential generated in PingOne while creating a PingIntelligence connection is assigned here so that PingOne can authenticate the data sent by ASE during runtime. For more information on PingOne connections, see Connections.
`abs_cloud_endpoint`	Use this parameter to assign an endpoint other than the one decoded by the gateway credentials. It's used when ABS is deployed in `cloud` mode.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`abs_endpoint`	The parameter has two possible configurations: • When ABS is deployed with a load balancer - Configure the host name and port or the IPv4 and port of the load balancer. **Note** To allow the load balancer to bind ASE's session to a specific ABS node, enable cookie stickiness in the load balancer with `PISESSIONID` cookie. • When ABS is deployed without a load balancer -Configure the parameter with the host name and port or the IPv4 and port of all the ABS nodes in a cluster. If later a new ABS node is added to the cluster, add its host name or IPv4 to the list and restart ASE.
`access_key`	The access key or the username for the ABS nodes. It is the same for all the ABS nodes. The same value has to be configured in ABS MongoDB database. This value is obfuscated during the start of ASE. This parameter is used when ABS is deployed in `onprem` mode. **Note** ":" is a restricted character and allowed in access key.
`secret_key`	The secret key or the password for the ABS nodes. It is the same for all the ABS nodes. The same value has to be configured in ABS MongoDB database. This value is obfuscated during the start of ASE. This parameter is used when ABS is deployed in `onprem` mode. **Note** ":" is a restricted character and allowed in secret key.
`enable_ssl`	Set the value to true for SSL communication between ASE and ABS. The default value is `true`. ASE sends the access log files in plain text if the value is set to `false`. This parameter is used when ABS is deployed in `onprem` mode.
`abs_ca_cert_path`	Location of the trusted certificate authority (CA) certificates for SSL/TLS connections from ASE to ABS. If the path parameter value is left empty, then ASE does not verify the validity of CA certificates. However, the connection to ABS is still encrypted. This parameter is used when ABS is deployed in `onprem` mode.

The `access_key` and `secret_key` are configured in ABS. For more information, see *ABS Admin Guide*.

The following is a sample `abs.conf` file:

Copyright © 2025 Ping Identity Corporation

```
; API Security Enforcer ABS configuration.;
 This file is in the standard .ini format.
The comments start with a semicolon (;).;
Following configurations are applicable only if ABS is enabled with true.
; Configure ABS deployment type. Supported values (onprem/cloud)
deployment_type=onprem

; PingIntelligence Gateway Credentials
gateway_credential=

; ABS endpoint for cloud
abs_cloud_endpoint=

; a comma-separated list of abs nodes having hostname:port or ipv4:port as an address.
abs_endpoint=127.0.0.1:8080

; access key for abs node
access_key=OBF:AES://ENOzsqOEhDBWLDY+pIoQ:jN6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0

; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw:zzi77GIFF5xkQJccjIrIVWU+RY5CxUhp3NLcNBel+3Q

; Setting this value to true will enable encrypted communication with ABS.
enable_ssl=true

; Configure the location of ABS's trusted CA certificates. If empty, ABS's certificate
; will not be verified
abs_ca_cert_path=
```

Configuring ASE to ABS encrypted communication

Steps

1. To enable SSL communication between ASE and ABS so that the access logs are encrypted and sent to ABS, set the value of `enable_ssl` to `true`.

   > **Note**
   >
   > The `abs_ca_cert_path` is the location of ABS AI engine's trusted CA certificate.

   > **Important**
   >
   > If the field is left empty, ASE does not verify ABS AI engine's certificate, but the communication is still encrypted.

2. Check and open ABS ports.

   The default port for connection with ABS is `8080`.

   1. To determine ABS accessibility, run the `check_ports.sh` script on the ASE machine.

Copyright © 2025 Ping Identity Corporation

Example:

```
/opt/pingidentity/ase/util ./check_ports.sh {ABS IPv4:[port]}
```

2. Input ABS host IP address and ports as arguments.

Managing ASE blocking of ABS-detected attacks

You can configure the API Security Enforcer (ASE) to automatically fetch and block API Behavioral Security (ABS)-detected attacks.

Enable or disable attack list fetching from ABS:

Enabling attack list fetching from ABS

Steps

1. To enable ASE Security, run the following command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
```

2. To enable ASE to send API traffic information to ABS, run the following command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs
```

3. To enable ASE to fetch and block ABS detected attacks, run the following command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs_attack
```

Result:

After enabling automated attack blocking, ASE periodically fetches the attack list from ABS and blocks the identified connections.

4. To set the time interval at which ASE fetches the attack list from ABS, configure the `abs_attack_request_minute` parameter in `ase.conf` file

Example:

```
; This value determines how often ASE will query ABS.
abs_attack_request_minutes=10
```


Copyright © 2025 Ping Identity Corporation

Disabling attack list fetching from ABS

Steps

- To disable ASE from fetching the ABS attack list, run the following command-line interface (CLI) command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs_attack
```

> **Note**
>
> The ABS attack list can be viewed using ABS APIs and used to manually configured an attack list on ASE. For more information on ABS APIs, see ABS Administration.

Result:

After entering the above command, ASE will no longer fetch the attack list from ABS. However, ABS continues generating the attack list and stores it locally.

- To stop an ASE cluster from sending log files to ABS, run the following ASE CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs
```

> **Note**
>
> For more information on types of attacks, see ABS AI Engine.

Result:

After entering this command, ABS will not receive any logs from ASE.

Configuring Google Pub/Sub

Google Cloud Pub/Sub is an enterprise event-driven message system. API Security Enforcer (ASE) integrates with Google Pub/Sub in ASE sideband mode. When you enable Google Pub/Sub in the `ase.conf` file, ASE sends the event message in a JavaScript Object Notation (JSON) file to Google Cloud.

Before you begin

You can verify that Google Pub/Sub is enabled by running the `status` command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh status -u admin -p admin
API Security Enforcer
status                  : started
mode                    : sideband
http/ws                 : port 80
https/wss               : port 443
firewall                : enabled
abs                     : disabled, ssl: enabled
abs attack              : disabled
audit                   : enabled
sideband authentication : disabled
ase detected attack     : disabled
attack list memory      : configured 128.00 MB, used 25.60 MB, free 102.40 MB
google pubsub           : enabled
```

About this task

Configure Google Pub/Sub in ASE:

Steps

1. Download the key file in JSON format from your Google Pub/Sub account.

   For more information on generating the key file, see Quickstart: building a functioning Cloud Pub/Sub system.

2. Copy the downloaded Key JSON file to `/pingidentity/ase/config` directory.
3. Rename the file to `google_application_credentials.json`.
4. Configure the following Google Pub/Sub options in the `ase.conf` file.

`enable_google_pubsub`	Set it to `true` if you want ASE to push metrics data to Google cloud. The default value is `false`. [NOTE] ==== ASE must be in the `sideband` mode for Google Pub/Sub configuration to take effect. ====
`google_pubsub_topic`	The path to your topic for publishing and subscribing the messages. For example, `/pingidentity/topic/your_topic`.
`google_pubsub_concurrency`	The number of concurrent connection between ASE and Google Pub/Sub. The maximum value is 1024 connections. The default value is 1000 connections.
`google_pubsub_qps`	The number of messages per second that ASE can publish to the topic. Maximum value is 10,000. The default value is 1000.
`google_pubsub_apikey`	The application programming interface (API) Key to establish connection between ASE and Google Pub/Sub. Configuring the API Key for Google Pub/Sub is optional.

Copyright © 2025 Ping Identity Corporation

`cache_queue_size`	The number of messages that are buffered in cache when ASE cannot publish to Google Pub/Sub. Maximum size of the queue is 10,000 messages. The default value is 300 messages.
`google_pubsub_timeout`	The time in seconds for which ASE tries to publish messages to Google Pub/Sub. In case of failure to publish, ASE makes three attempts to publish the message, after which it writes the message to the `google_pubsub_failed.log` file.

Result

ASE sends the event information to Google Pub/Sub in a JSON message. The message captures the following information:

- Method
- URL
- Host
- Request time-stamp
- Request length
- Source IP
- X-forwarded-for IPs
- Response code
- Response length, and
- Latency in milliseconds

ASE makes three attempts to publish the message to Google Pub/Sub, after which the entire message is logged in failed log file. The message that is logged in the failed log file is not in plain text. If the message is not published to Google Pub/Sub, you can check the reason for failure in `balancer.log` file. For more information on `balancer.log` file, see ASE access, management, and audit logs.

When messages are successfully published to Google Pub/Sub, the message ID is logged in the success log file. The following is a snippet of an event message JSON file logged in `balancer.log` file when ASE is running in debug mode:

Copyright © 2025 Ping Identity Corporation

```
{
  "method": "PUT",
  "url": "/shopapi-books/order",
  "host": "shop-electronics.cloudhub.io",
  "request_timestamp": "1573767522429",
  "request_length": "464",
  "source_ip": "1.2.3.4",
  "x_forwarded_for": "1.1.1.1, 1.1.1.2",
  "response_code": "200",
  "response_length": "26",
  "latency_ms": "208"
}
```


Configuring the API key

Before you begin

Obtain the API key for your Google project and configure it in the `google_pubsub_apikey` option. Obfuscate the API key for it to take effect. For more information on obfuscating keys and password, see Obfuscating keys and passwords.

About this task

You can optionally configure the API key in the `ase.conf` file.

Steps

1. Stop ASE.

   For more information, see Stopping ASE.

2. Edit the `ase.conf` file to add the API key.

3. Obfuscate the API key.

4. Start ASE.

   For more information, see Starting ASE.

REST APIs for sideband token and authentication

API Security Enforcer (ASE) provides REST application programming interface (API)s for authentication and sideband token management.

The Authentication REST API

The `Authentication` API helps to enable and disable ASE sideband authentication. You can also retrieve the authentication status.

Copyright © 2025 Ping Identity Corporation

Enable or disable sideband authentication

URL	`/v5/ase/sideband/authentication?status=<value>`	
Method	POST	
Query Parameter	`status`	Valid values:*enable* or *disable*
Request Headers	**x-ase-access-key: <value>** **x-ase-secret-key: <value>**	

The following is a sample curl command:

```
curl --location --request POST '<ASE IP Address>:<port no>/v5/ase/sideband/authentication?status=enable
' \
--header 'x-ase-access-key: ase_ak' \
--header 'x-ase-secret-key: ase_sk'
```

The following are sample responses:

```
{
    "status": "disabled",
    "status_message": "Sideband authentication is disabled"
}
```

```
{
    "status": "enabled",
    "status_message": "Sideband authentication is enabled"
}
```

Get authentication status

URL	**/v5/ase/sideband/authentication**
Method	GET

Copyright © 2025 Ping Identity Corporation

Request Headers	`x-ase-access-key: <value>` `x-ase-secret-key: <value>`

The following is a sample curl command:

```
curl --location --request POST '<ASE IP Address>:<port no>/v5/ase/sideband/authentication' \
--header 'x-ase-access-key: ase_ak' \
--header 'x-ase-secret-key: ase_sk'
```

The following are sample responses:

```
{
"status": "disabled",
"status_message": "Sideband authentication is disabled"
}
```

```
{
"status": "enabled",
"status_message": "Sideband authentication is enabled"
}
```

The Token REST API

The `Token` API helps to create, import, and delete ASE sideband tokens. You can also retrieve the list of tokens issued by ASE.

Create a new sideband token

URL	`/v5/ase/sideband/token`
Method	POST
Request Headers	`x-ase-access-key: <value>` `x-ase-secret-key: <value>`

The following is sample curl command:

Copyright © 2025 Ping Identity Corporation

```
curl --location --request POST '<ASE IP Address>:<port no>/v5/ase/sideband/token' \
--header 'x-ase-access-key: ase_ak' \
--header 'x-ase-secret-key: ase_sk'
```

The following is a sample response:

```
{
    "status": "token_created",
    "token": "dac5fkdfjdlfjdlfjldkfjd1ab08903453fec4c0"
}
```

Import a sideband token

The token should be 32 character long, and the allowable characters in the token are alphabets in lowercase and digits 0-9.

URL	`/v5/ase/sideband/token`
Method	PUT

Request Headers	`x-ase-access-key:  <value>`
	`x-ase-secret-key:  <value>`
	`Content-Type: application/json`

The following is a sample curl command:

```
curl --location --request PUT '<ASE IP Address>:<port no>/v5/ase/sideband/token' \
--header 'x-ase-access-key: admin' \
--header 'x-ase-secret-key: admin' \
--header 'Content-Type: application/json' \
--data-raw '{
  "token": "dc6684370f014923b8a070c982601f7c"
}
```

The following is a sample request:

```
{   "token": "dc6684370f014923b8a070c982601f75"}
```

The following is a sample response:

Copyright © 2025 Ping Identity Corporation

```
{
    "status": "success",
    "status_message": "Sideband token dc6684370f014923b8a070c982601f75 imported."
}
```

Delete a sideband token

URL	`/v5/ase/sideband/token`
Method	DELETE

Request Headers	`x-ase-access-key:  <value>`
	`x-ase-secret-key:  <value>`
	`Content-Type: application/json`

The following is a sample curl command:

```
curl --location --request DELETE '<ASE IP Address>:<port no>/v5/ase/sideband/token' \
--header 'x-ase-access-key: admin' \
--header 'x-ase-secret-key: admin' \
--header 'Content-Type: application/json' \
--data-raw '{
  "token": "dc6684370f014923b8a070c982601f7c"
}
```

The following is a sample request:

```
{   "token": "dc6684370f014923b8a070c982601f75"}
```

The following is a sample response:

```
{
    "status": "success",
    "status_message": "Sideband token dc6684370f014923b8a070c982601f75 deleted."
}
```


Copyright © 2025 Ping Identity Corporation

List all sideband tokens

URL	`/v5/ase/sideband/token`
Method	GET
Request Headers	`x-ase-access-key: <value>` `x-ase-secret-key: <value>`

The following is a sample curl command:

```
curl --location --request GET '<ASE IP Address>:<port no>/v5/ase/sideband/token' \
--header 'x-ase-access-key: ase_ak' \
--header 'x-ase-secret-key: ase_sk'
```

The following is a sample response:

```
{
    "sideband_tokens": [
                {
            "token": "dac559bf75904141ab08903453fec4c0",
            "created_at": "2021-May-06 14:02:44"
        },
        {
            "token": "dc6684370f014923b8a070c982601c74",
            "created_at": "2021-May-06 13:51:55"
        }
    ]
}
```

CLI for sideband ASE

The following table shows the command-line interface (CLI) functions and their syntax for API Security Enforcer (ASE) in sideband mode.

Function	Description	Syntax
Start ASE	Starts ASE	`./start.sh`
Stop ASE	Stops ASE	`./stop.sh`
Help	Displays cli.sh help	`./cli.sh help`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
Version	Displays the version number of ASE	`./cli.sh version`
Status	Displays the running status of ASE	`./cli.sh status`
Update Password	Changes ASE admin password	`./cli.sh update_password -u admin - p`
Change log level	Changes `balancer.log` and `controller.log` log level	`./cli.sh log_level -u admin -p` Options: • `warn` • `info` • `error` • `fatal` • `debug`
Get Authentication Method	Displays the current authentication method	`./cli.sh get_auth_method -u admin -p`
Update Authentication Method	Updates ASE authentication method	`./cli.sh update_auth_method {method} -u admin -p`
Enable Sideband Authentication	Enables authentication between API gateway and ASE when ASE is deployed in sideband mode	`./cli.sh enable_sideband_authentication -u admin – p`
Disable Sideband Authentication	Disables authentication between API gateway and ASE when ASE is deployed in sideband mode	`./cli.sh disable_sideband_authentication -u admin – p`
Create ASE Authentication Token	Create the ASE token that is used to authenticate between the API gateway and ASE	`./cli.sh create_sideband_token -u admin – p`
List ASE Authentication Token	List the ASE token that is used to authenticate between the API gateway and ASE	`./cli.sh list_sideband_token -u admin – p`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
Import ASE Authentication Token	Imports ASE token that is used for authentication between ASE and API gateway. The token should be 32 characters long and the allowable characters in the token are alphabets in small case and digits 0-9.	`./cli.sh import_sideband_token {token} -u admin – p admin`
Delete ASE Authentication Token	Deletes the ASE token that is used to authenticate between the API gateway and ASE	`./cli.sh delete_sideband_token {token} -u admin – p`
Enable Audit Logging	Enables audit logging	`./cli.sh enable_audit -u admin -p admin`
Disable Audit Logging	Disables audit logging	`./cli.sh disable_audit -u admin -p admin`
Add Syslog Server	Adds a new syslog server	`./cli.sh –u admin -p admin add_syslog_server host:port`
Delete Syslog Server	Deletes the syslog server	`./cli.sh –u admin -p admin delete_syslog_server host:port`
List Syslog Server	Lists the current syslog server	`./cli.sh –u admin -p admin list_syslog_server`
Add API	Add a new API file in JSON format. File should have `.json` extension. Provide the complete path where you have stored the API JSON file. After running the command, API is added to `/opt/pingindentity/ase/config/api` directory	`./cli.sh –u admin -p admin add_api {config_file_path}`
Update API	Updates an API after the API JSON file has been edited and saved	`./cli.sh –u admin -p admin update_api {api_name}`
List APIs	Lists all APIs configured in ASE	`./cli.sh –u admin -p admin list_api`
API Info	Displays the API JSON file	`./cli.sh –u admin -p admin api_info {api_id}`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
API Count	Displays the total number of APIs configured	`./cli.sh –u admin -p admin api_count`
Enable Per API Blocking	Enables attack blocking for the API	`./cli.sh –u admin -p admin enable_blocking {api_id}`
Disable Per API Blocking	Disables attack blocking for the API	`./cli.sh –u admin -p admin disable_blocking {api_id}`
Delete API	Deletes an API from ASE. Deleting an API removes the corresponding JSON file and deletes all the cookies associated with that API	`./cli.sh –u admin -p admin delete_api {api_id}`
Generate Master Key	Generates the master obfuscation key `ase_master.key`	`./cli.sh -u admin -p admin generate_obfkey`
Obfuscate Keys and Password	Obfuscates the keys and passwords configured in various configuration files	`./cli.sh -u admin -p admin obfuscate_keys`
Create a Key Pair	Creates private key and public key pair in keystore	`./cli.sh –u admin -p admin create_key_pair`
Create a CSR	Creates a certificate signing request	`./cli.sh –u admin -p admin create_csr`
Create a Self-Signed Certificate	Creates a self-signed certificate	`./cli.sh –u admin -p admin create_self_sign_cert`
Import Certificate	Imports a CA-signed certificate into keystore	`./cli.sh –u admin -p admin import_cert {cert_path}`
Create Management Key Pair	Creates a private key for management server	`/cli.sh –u admin -p admin create_management_key_pair`
Create Management CSR	Creates a certificate signing request for management server	`/cli.sh –u admin -p admin create_management_csr`
Create Management Self-signed Certificate	Creates a self-signed certificate for management server	`/cli.sh –u admin -p admin create_management_self_sign_cert`
Import Management Key Pair	Imports a key-pair for management server	`/cli.sh –u admin -p admin import_management_key_pair {key_path}`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
Import Management Certificate	Imports a CA-signed certificate for management server	`/cli.sh –u admin -p admin import_management_cert {cert_path}`
Cluster Info	Displays information about an ASE cluster	`./cli.sh –u admin -p admin cluster_info`
Delete Cluster Node	Deletes an inactive ASE cluster node	`./cli.sh –u admin -p admin delete_cluster_node host:port`
Enable Firewall	Enables API firewall. Activates pattern enforcement, API name mapping, manual attack type	`./cli.sh –u admin -p admin enable_firewall`
Disable Firewall	Disables API firewall	`./cli.sh –u admin -p admin disable_firewall`
Enable ASE detected attacks	Enables ASE-detected attacks	`./cli.sh –u admin -p admin enable_ase_detected_attack`
Disable ASE Detected Attacks	Disables ASE-detected attacks	`./cli.sh –u admin -p admin disable_ase_detected_attack`
Enable ABS	Enables ABS to send access logs to ABS	`./cli.sh –u admin -p admin enable_abs`
Disable ABS	Disables ABS to stop sending access logs to ABS	`./cli.sh –u admin -p admin disable_abs`

Adding deny list	Adds an entry to ASE deny list using CLI. Valid type values are: IP, Cookie, OAuth2 token, API Key, and username	`./cli.sh –u admin -p admin add_blacklist {type}{name}{value}`
		If type is `ip`, then `name` is the IP address.
		If type is `cookie`, then `name` is the cookie name, and `value` is the cookie value
		Example:
		`/cli.sh -u admin -p admin add_blacklist ip 1.1.1.1`

Delete deny list Entry	Deletes entry from the deny list.	`./cli.sh –u admin -p admin delete_blacklist {type}{name}{value}`
		Example:
		`cli.sh -u admin -p delete_blacklist token 58fcb0cb97c54afbb88c07a4f2d73c35`

Clear deny list	Clears all the entries from the deny list	`./cli.sh –u admin -p admin clear_blacklist`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
View deny list	Views the entire deny list or view a deny list for the specified attack type (for example, invalid_method)	`./cli.sh –u admin -p admin view_blacklist \{all|manual|abs_generated|invalid_content_type|invalid_method|invalid_protocol|decoy|missing_token}`
View deny list for IP addresses with missing tokens	Views the deny list entries that are blocked due to missing tokens	`./cli.sh view_blacklist missing_token -uadmin -padmin`

Adding allow list	Adds an entry to ASE allow list using CLI. Valid type values are: IP, cookie, OAuth2 token, API key, and username	`./cli.sh –u admin -p admin add_whitelist {type}{name}{value}`
		Options:
		• If type is `ip`, then `name` is the IP address.
		• If type is `cookie`, then `name` is the cookie name, and `value` is the cookie value
		Example:
		`/cli.sh -u admin -p admin add_whitelist api_key AccessKey 065f73cdf39e486f9d7cda97d2dd1597`

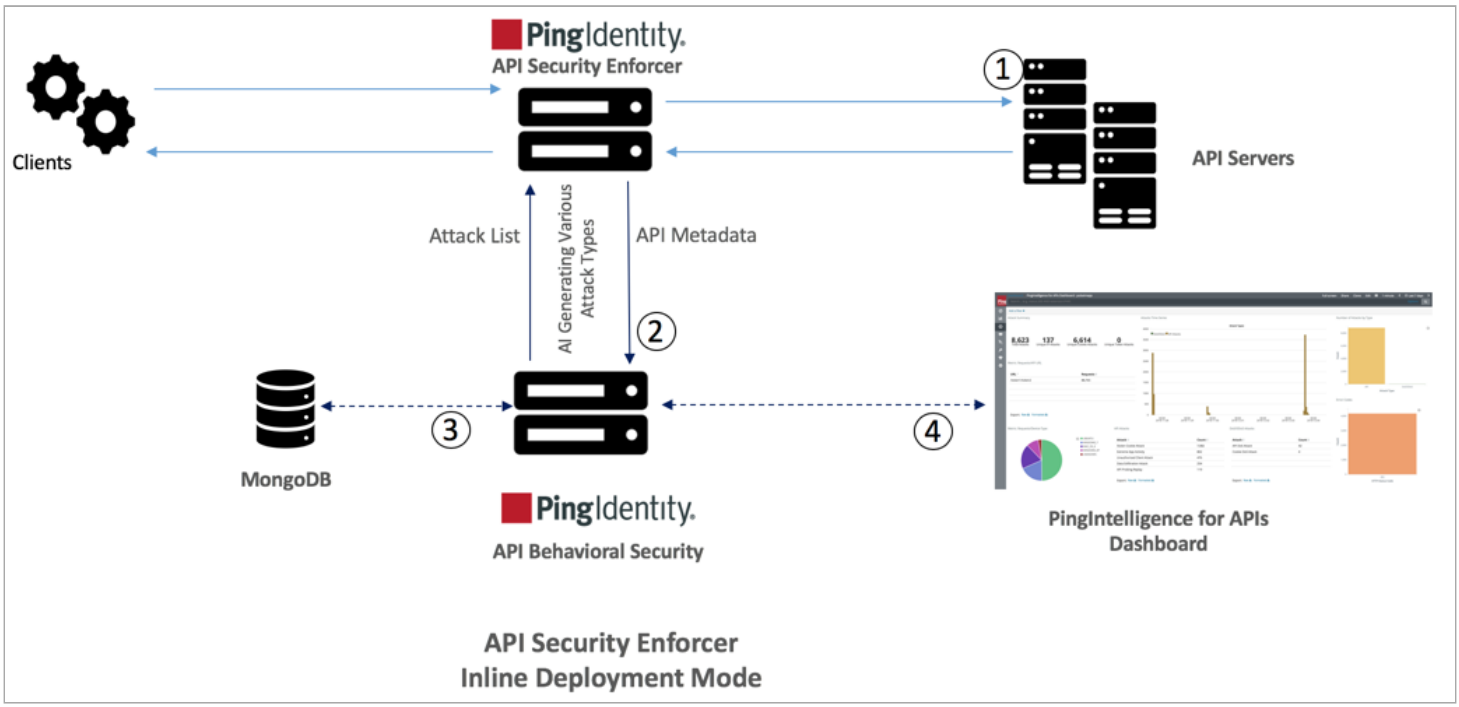
Delete allow list Entry	Delete entry from the allow list	`./cli.sh –u admin -p admin delete_whitelist {type}{name}{value}` Example: **`/cli.sh -u admin -p delete_whitelist token 58fcb0cb97c54afbb88c07a4f2d73c35`**
Clear allow list	Clears all the entries from the allow list	`./cli.sh –u admin -p admin clear_whitelist`
View allow list	Views the entire allow list	`./cli.sh –u admin -p admin view_whitelist`
ABS Info	Displays ABS status information ABS enabled or disabled, ASE fetching ABS attack types, and ABS cluster information	`./cli.sh –u admin -p admin abs_info`

Inline ASE

In the inline deployment mode, API Security Enforcer (ASE) sits at the edge of your network to receive the application programming interface (API) traffic.

Copyright © 2025 Ping Identity Corporation

ASE can be deployed behind an existing load balancer such as Amazon Web Service (AWS) ELB. ASE deployed at the edge of the data center terminates SSL connections from API clients. It then routes the requests directly to the correct destination APIs and app servers such as Node.js, WebLogic, Tomcat, PHP, etc.



To configure ASE to work in Inline mode, set the `mode=inline` in the `ase.conf` file.

Some load balancers (for example, Amazon Web Services (AWS) ELB) require responses to keep alive messages from all devices receiving traffic. In an inline mode configuration, ASE should be configured to respond to these keep alive messages by updating the `ase_health` variable in the `ase.conf` file. When `ase_health` is set to `true`, load balancers can perform an ASE health check using the following URL: `http(s)://<ASE Name>/ase`, where *<ASE Name>* is the ASE domain name. ASE will respond to these health checks.

Inline ASE configuration using the ase.conf file

API Security Enforcer (ASE) system level configuration entails modifying parameters in the `ase.conf` file located in the `config` directory. Some values have default settings which can be modified to support your application requirements.

The following table includes the parameter values and descriptions.

Parameter	Description
ASE mode	
`mode`	The mode in which ASE works. Possible values are `inline` and `sideband`. The default value is `inline`.
ASE timezone	

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`timezone`	Sets ASE's timezone. The values can be `local` or `UTC`. Default value is `UTC`. If ASE is deployed in a cluster, configure the same timezone on each cluster node manually.
`enable_sideband_keepalive`	N/A
`enable_sideband_authentication`	N/A
ASE ports	
`http_ws_port`	Data port used for http or WebSocket protocol. The default value is 8000.
`https_wss_port`	Data port used for https or Secure WebSocket (wss). The default value is 8443.
`management_port`	Management port used for command-line interface (CLI) and REST application programming interface (API) management. The default value is 8010.
ASE administration and audit	
`admin_log_level`	The level of log detail captured. Options include: Fatal – 1, Error – 2, Warning – 3, Info – 4, Debug – 5
`enable_audit`	When set to `true`, ASE logs all actions performed in ASE in the audit log files. The default value is `true`.
`syslog_server`	Syslog server hostname or `IPv4 address:port number`. Leave this parameter blank if you do not want to generate for no syslog.
`hostname_refresh`	Time interval at which hostnames are refreshed. The default value is 60 secs. When ASE attempts to refresh the hostname, the hostname resolution must happen in 5 secs.
`auth_method`	Authentication method used for administrator access. See Configuring Native and PAM Authentication for more information on the two options: • `ase::db` (Default - Native authentication) • `pam::ldap` (Linux-PAM Authentication with script)
`enable_ase_health`	When `true`, enables load balancers to perform a health check using the following URL: `http(s)://<ASE Name>/ase` where *<ASE Name>* is the ASE domain name The default value is `false`. **Note** Do not configure the /ase URL in an API JavaScript Object Notation (JSON) file.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`enable_1G`	**When `true`, enable 1Gbps Ethernet support.** **The default value is `true`.** **Note** Only applicable when using a 1G NIC card
`http_ws_process`	**The number of HTTP or WebSocket processes.** **The default value is 1 and the maximum value is 6.** **Note** When running ASE in a cluster deployment, all nodes must have the same number of processes.
`https_wss_process`	**The number of HTTPS or secure WebSocket processes.** **The default value is 1 and the maximum value is 6.** **Note** When running ASE in a cluster deployment, all nodes must have the same number of processes.
`enable_access_log`	**When `true`, log client traffic request and response information. Default value is `true`. Make sure the value is set to `true` when ASE connected to PingOne.**
`flush_log_immediate`	**When `true`, log files are immediately written to the file system. When `false`, log files are written after a time interval. The default value is `true`.**
`attack_list_memory`	**The amount of memory used for maintaining black and whitelists. The default value is 128 MB.**
`keystore_password`	**Password for the keystore. For more information on updating the keystore password, see Updating Keystore Password.**

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`enable_hostname_rewrite`	When set to `true`, ASE rewrites the host header in the client request with the Internet Protocol (IP) or host and port number configured in the `server` section of the API JSON. Make a note of the following points: server_ssl in API JSON set to `false`: • In the server section of API JSON, if the configured port is the standard HTTP port (port number 80), then only the IP or hostname in the request header is rewritten. • In the server section of API JSON, if the configured port is any port other than the standard HTTP port (port number 80), then IP or hostname and port number in the request header is rewritten. For example, if the configured port number is 8080 in API JSON for a host example.com, then ASE rewrites the host header in request with example.com:8080. server_ssl in API JSON set to `true`: • In the server section of API JSON, if the configured port is the standard HTTPS port (port number 443), then only the IP or hostname in the request header is rewritten. • In the server section of API JSON, if the configured port is any port other than the standard HTTPS port (port number 443), then IP or hostname and port number in the request header is rewritten. For example, if the configured port number is 8443 in API JSON for a host example.com, then ASE rewrites the host header in request with example.com:8443.
ASE cluster	
`enable_cluster`	When `true`, run the setup in cluster mode. The default value is `false`, run the setup in standalone mode.
Security	
`enable_sslv3`	When `true`, enable Secure Sockets Layer (SSL) 3. Default value is `false`.
`server_ca_cert_path`	Location of the trusted certificate authority (CA) certificates for SSL/TLS connections from ASE to backend servers. If the path parameter value is left empty, then ASE does not verify the validity of CA certificates. However, the backend connection is still encrypted. For RHEL 7.6 CA certificates, the default path is: `/etc/pki/tls/certs/.` Multiple certificates can be placed in this directory.
`enable_xff`	When `true`, pass XFF header with originating IP address to the backend server.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`enable_firewall`	When `true`, activate the following API security features: • API mapping • API pattern enforcement • Connection drop using attack types • Flow control Default value is `true`
`enable_strict_request_parser`	When `true`, ASE blocks client http requests with invalid headers start. The default value is `true`.
Real-time API security	
`enable_ase_detected_attack`	When `true`, activates the real-time security in ASE. ASE detects and blocks pattern enforcement violations, wrong API keys and clients probing decoy API and later accessing real APIs. The default value is `false`.
API deception	
`decoy_alert_interval`	The time interval between decoy API email alerts. The default value is 180 minutes. Maximum value is 1440 minutes (i.e. 24 hours).
AI-based API Behavioral Security (ABS)	
`enable_abs`	When `true` (default), send access log files to ABS AI Engine for generating API metrics and detecting attacks using machine learning algorithms. Set it to `true` when ASE is connected to PingOne.
`enable_abs_attack`	When `true` (default), ASE fetches attack list from ABS AI Engine and blocks access by the clients that are in the attack list. When `false`, attack list is not downloaded.
`abs_attack_request_minute`	Time interval in minutes at which ASE fetches ABS attack list. The default value is 10-minutes.
Google Pub/Sub configuration	
`enable_google_pubsub`	N/A
`google_pubsub_topic`	N/A
`google_pubsub_concurrency`	N/A
`google_pubsub_qps`	N/A

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`google_pubsub_apikey`	N/A
`cache_queue_size`	N/A
`google_pubsub_timeout`	N/A
API Publish (ABS)	
`enable_abs_publish`	When `true`, ASE polls ABS to get list of published APIs and list of non-discovered APIs and decide whether APIs received will be added, deleted or updated. When `false`, the published list will not be downloaded. The default value is `false`.
`abs_publish_request_minutes`	This value determines how often ASE will get published API list from ABS. The default value is `10 minutes`.
Alerts and reports	
`enable_email`	When `true`, send email notifications. The default value is `false`. ASE logs the alerts in `balancer.log` file even when email alerts are disabled. See Email alerts and reports for more information.
`email_report`	Time interval in days at which ASE sends reports. Minimum value is 1 day and the maximum is 7-days. The default value is 1-day.
`smtp_host`	Hostname of SMTP server.
`smtp_port`	Port number of SMTP server.
`smtp_ssl`	Set to `true` if you want email communication to be over SSL. Make sure that the SMTP server supports SSL. If you set `smtp_ssl` to `true` and the SMTP server does not support SSL, email communication falls back to the non-SSL channel. The default value is `true`. Set it to `false` if email communication is over a non-SSL channel. The email communication will fail if you set the parameter to `false`, but the SMTP server only supports SSL communication.

Copyright © 2025 Ping Identity Corporation

Parameter	Description

`smtp_cert_verification`	**Set to `true` if you want ASE to verify the SMTP server's SSL certificate. The default value is `true`.**
	If you set it to `false`, ASE does not verify SMTP server's SSL certificate; however, the communication is still over SSL.
	Note
	If you have configured an IP address as `smtp_host` and set `smtp_cert_verification` to `true`, then make sure that the certificate configured on the SMTP server has the following:
	`X509v3 extensions:`
	`X509v3 Key Usage:`
	`Key Encipherment, Data Encipherment`
	`X509v3 Extended Key Usage:`
	`TLS Web Server Authentication`
	`X509v3 Subject Alternative Name:`
	`IP Address: X.X.X.X`

`sender_email`	**Email address for sending email alerts and reports.**
`sender_password`	**Password of sender's email account.** **Note** You can leave this field blank if your SMTP server does not require authentication.
`receiver_email`	**Email address to notify about alerts and reports** **See email alerts for more information.**
ASE server resource utilization	
`cpu_usage`	**Percentage threshold value of CPU utilization.** **See email alerts for more information.**
`memory_usage`	**Percentage threshold value of memory usage.** **See email alerts for more information.**
`filesystem_size`	**Percentage threshold value of filesystem capacity.** **See email alerts for more information.**
`buffer_size`	**Customizable payload buffer size to reduce the number of iterations required for reading and writing payloads.** **Default value is 16KB. Minimum is 1KB and maximum is 32KB.**

A sample `ase.conf` file is displayed below:

Copyright © 2025 Ping Identity Corporation

```
; This is API Security Enforcer's main configuration file. This file is in the standard .ini format.
; It contains ports, firewall, log, ABS flags. The comments start with a semicolon (;).

; Defines running mode for API Security Enforcer (Allowed values are inline or sideband).
mode=inline

; Defines http(s)/websocket(s) ports for API Security Enforcer. Linux user should have the privilege to
bind to these ports.
; If you comment out a port, then that protocol is disabled.
http_ws_port=8000
https_wss_port=8443

; REST API
management_port=8010

; For controller.log and balancer.log only
; 1-5 (FATAL, ERROR, WARNING, INFO, DEBUG)
admin_log_level=4

; Defines the number of processes for a protocol.
; The maximum number of allowed process for each protocol is 6 (1 master + 5 child). The
; following defines 1 process for both http/ws and https/wss protocol.
http_ws_process=1
https_wss_process=1

; Enable or disable access logs to the filesystem (request/response).
; WARNING! It must be set to true for sending logs to ABS for analytics.
enable_access_log=true
; To write access log immediately to the filesystem, set to true.
flush_log_immediate=true

; Setting this value to true will enable this node to participate in an API Security Enforcer
; cluster. Define cluster configurations in the cluster.conf
enable_cluster=false

; Current API Security Enforcer version has 3 firewall features: API Mapping, API Pattern
; Enforcement, and Attack Types.
enable_firewall=true

; X-Forwarded For
enable_xff=false

; SSLv3
enable_sslv3=false

; enable Nagle's algorithm (if NIC card is 1G).
enable_1G=true

; tcp send buffer size in bytes(kernel)
tcp_send_buffer_size=65535
; tcp receive buffer size in bytes(kernel)
tcp_receive_buffer_size=65535

; buffer size for send and receive in KBs (user)
```


Copyright © 2025 Ping Identity Corporation

```
buffer_size=16KB

; Set this value to true, to allow API Security Enforcer to send logs to ABS. This
; configuration depends on the value of the enable_access_log parameter.
enable_abs=true

; Set this value to true, to allow API Security Enforcer to fetch attack list from ABS.
enable_abs_attack=true

; This value determines how often API Security Enforcer will get attack list from ABS.
abs_attack_request_minutes=10

; Set this value to true, to allow API Security Enforcer to fetch published API list from ABS.
enable_abs_publish=false

; This value determines how often API Security Enforcer will get published API list from ABS.
abs_publish_request_minutes=10

; Set this value to true, to allow API Security Enforcer to block auto detected attacks.
enable_ase_detected_attack=false

; Set this value to true to enable email for both alerts and daily reports.
enable_email=false

; Defines report frequency in days [0=no reports, 1=every day, 2=once in two days and max is 7 ; days]
email_report=1
; Specify your email settings
smtp_host=smtp://<smtp-server>
smtp_port=587
; Set this value to true if smtp host support SSL
smtp_ssl=true
; Set this value to true if SSL certificate verification is required
smtp_cert_verification=false
sender_email=
sender_password=
receiver_email=

; Defines threshold for an email alert. For example, if CPU usage is 70%, you will get an
; alert.
cpu_usage=70
memory_usage=70
filesystem_size=70

; Authentication method. Format is <auth_agent>::<auth_service>
; Valid values for auth_agent are ase and pam
; ase agent only supports db auth_service
; pam agent can support user configured pam services
; For example ase::db, pam::passwd, pam::ldap etc
auth_method=ase::db

; Enable auditing. Valid values are true or false.
enable_audit=true

; Decoy alert interval in minutes. [min=15, default=3*60, max=24*60]
decoy_alert_interval=180
```

Copyright © 2025 Ping Identity Corporation

```
; Interval for a hostname lookup (in seconds). [min=10, default=60, max=86400]
hostname_refresh=60

; Syslog server settings. The valid format is host:port. Host can be an FQDN or an IPv4
; address.
syslog_server=

; Attack List size in MB or GB. [min=64MB, max=1024GB]
; ASE will take 3*(configured memory) internally. Make sure that the system has at least
; 3*(configured memory) available
; If you are running ASE inside a container, configure the container to use 3*(configured
; memory) shared memory.
attack_list_memory=128MB

; Enable or Disable health check module. ASE uses '/ase' url for both http and https. This is
; useful if ASE is deployed behind a load balancer.
enable_ase_health=false

; Location for server's trusted CA certificates. If empty, Server's certificate will not be
; verified.
server_ca_cert_path=

; enable client side authentication. This setting is applicable only in sideband mode. Once enabled
; request will be authenticated using authentication tokens.
enable_sideband_authentication=false

; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false

; keystore password
keystore_password=OBF:AES:sRNp0W7sSi1zrReXeHodKQ:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerMPwvM4

; enable hostname rewrite for inline mode. ASE will rewrite the host header in request
; to the server's hostname
enable_hostname_rewrite=false

; enable strict parsing checks for client requests
; If enabled, ASE will block request with invalid header start
; If disabled, it will allow requests
; default value = true
enable_strict_request_parser=true

; Set the timezone to utc or local. The default timezone is utc.
timezone=utc

; Google Pub Sub Configuation
enable_google_pubsub=false

google_pubsub_topic=/topic/apimetrics

; Number of concurrent connections to Google Pub/Sub
```


Copyright © 2025 Ping Identity Corporation

```
; Minimum: 1, Default: 1000, Maximum: 1024
google_pubsub_concurrency=1000

; Number of messages published per second.
; Minimum: 1, Default: 1000, Maximum: 10000
google_pubsub_qps=1000

; Google service account API key (Optional)
google_pubsub_apikey=

; Maximum number of messages buffered in memory
; If queue is full, messages are written to logs/google_pubsub_failed.log
; Minimum: 1, Default: 300, Maximum: 10000
cache_queue_size=300

; Timeout in seconds to publish a message to Google Pub/Sub.
; Minimum: 10, Default: 30, Maximum: 300
google_pubsub_timeout=30
```

API naming guidelines

The application programming interface (API) name must follow the following guidelines.

- The name should not have the word “model”.
- The name should not have the word “threshold”.
- The name should not have the word “all”.
- The name should not have the word “decoyall”.

The following is the list of allowed characters in API name:

- The maximum characters in API name can be 160
- - (hyphen), _ (underscore), and white space are allowed in the name
- a-z, A-Z, and 0-9
- The first character must be alphanumeric

Defining an API using API JSON configuration file in inline mode

The application programming interface (API) JavaScript Object Notation (JSON) file parameters define the behavior and properties of your API.

The sample API JSON files shipped with API Security Enforcer (ASE) can be changed to your environment settings and are populated with default values.

The following table describes the JSON file parameters.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`protocol`	API request type with supported values of: • `ws` - WebSocket • `http` - HTTP
`url`	The value of the URL for the managed API. You can configure up to six levels of sub-paths. For example: • `/shopping` - name of a 1 level API • `/shopping/electronics/phones/brand` - 4 level API • `/` - entire server. Used for API Behavioral Security (ABS) API Discovery or load balancing.
`hostname`	Host name for the API. The value cannot be empty. `"*"` matches any host name.
`cookie`	Name of cookie used by the backend servers.
`cookie_idle_timeout`	The amount of time a cookie is valid, such as 20m for 20 minutes. The time duration formats include: • s - seconds • m - minutes • h - hour • d - day • w - week • mnt - month • yr - year
`logout_api_enabled`	When `true`, ASE expires cookies when a logout request is sent.
`cookie_persistence_enabled`	When `true,` the subsequent request from a client is sent to the server which initially responded.
`oauth2_access_token`	When `true`, ASE captures OAuth2 Access Tokens. When `false`, ASE does not look for OAuth2 Tokens. Default value is `false`. For more information, see Capture client identifiers in inline mode.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`is_token_mandatory`	When set to `true`, if the request has a missing token, ASE adds the IP address of the client to deny list and blocks the request. When set to `false`, ASE does not block the client. **Important** For ASE to check and block the client the following values must be set to `true`: • `oauth2_access_token` • `enable_firewall` and `enable_ase_detected_attack` in Inline ASE configuration using the ase.conf file. The default value is `false`.
`apikey_qs`	When API Key is sent in the query string, ASE uses the specified `parameter name` to capture the API key value. For more information, see Capture client identifiers in inline mode.
`apikey_header`	When API Key is part of the header field, ASE uses the specified parameter name to capture the API key value. For more information, see Capture client identifiers in inline mode.
`login_url`	Public URL used by a client to connect to the application.
`enable_blocking`	When `true`, ASE blocks all types of attack on this API. When `false`, no attacks are blocked. Default value is `false`.
`api_memory_size`	Maximum ASE memory allocation for an API. The default value is 128 MB. The data unit can be MB or GB.
`health_check`	When `true`, enable health checking of backend servers. When `false`, no health checks are performed. Ping Identity recommends setting this parameter as `true`.
`health_check_interval`	The interval in seconds at which ASE sends a health check to determine backend server status.
`health_retry_count`	The number of times ASE queries the backend server status after not receiving a response.
`health_url`	The URL used by ASE to check backend server status.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`health_check_headers`	Configure one or more health check headers in the API JSON in a key-value format. This is an optional configuration and applies only to inline ASE deployment. In the sample JSON, the following example is provided: `"health_check_headers": { "X-Host": "%{HOST}", "X-Custom-Header": "value" },` The next table explains the sample JSON.
`server_ssl`	When set to `true`, ASE connects to the backend API server over Secure Sockets Layer (SSL). If set to `false`, ASE uses TCP to connect to the backend server.

Servers:
`host` `port` `server_spike_threshold`
`server_connection_quota`
	The IP address or hostname and port number of each backend server running the API.
	See REST API Protection from DoS and DDoS for information on optional flow control parameters.

API Mapping: `internal_url`	Internal URL is mapped to the public external URL See API Name Mapping – Protect Internal URLs for more information
The following API Pattern Enforcement parameters only apply when API Firewall is activated	

Flow Control
`client_spike_threshold`
`server_connection_queueing`
`bytes_in_threshold`
`bytes_out_threshold`
	ASE flow control ensures that backend API servers are protected from surges (for example DDoS, traffic spike) in API traffic.
	See WebSocket API Protection from DoS and DDoS for information on parameters.

`protocol_allowed`	List of accepted protocols Values can be HTTP, HTTPS, WS, WSS. **Note** When Firewall is enabled, `protocol_allowed` takes precedence over the `protocol` parameter.

`http_redirect`
`response_code`
`response_def`
`https_url`
	Redirect unencrypted HTTP requests to `http_redirect`, the FQDN address of a HTTPS secure connection.
	See Configuring Pattern Enforcement for details.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`methods_allowed`	List of accepted REST API methods. Possible values are: • `GET` • `POST` • `PUT` • `DELETE` • `HEAD`
`content_type_allowed`	List of content types allowed. Multiple values cannot be listed. For example, application/json.

`error_code`
`error_type`
`error_message_body`
	Error message generated by ASE after blocking a client.
	See ASE Detected Error Messages for details.

Decoy Config `decoy_enabled` `response_code` `response_def` `response_message` `decoy_subpaths`	When `decoy_enabled` is set to `true`, decoy sub-paths function as decoy APIs. response_code The status code (for example, 200) that ASE returns when a decoy API path is accessed. response_def The response definition (for example OK) that ASE returns when a decoy API path is accessed. response_message The response message (for example OK) that ASE returns when a decoy API path is accessed. decoy_subpaths The list of decoy API sub-paths (for example `shop/admin`, `shop/root)`. For more information see Configuring API deception for details.
`username_header`	The name of the custom header containing username. When the value of `username_header` is set, ASE extracts the username from the custom header. For more information, see Extract username from custom header in inline mode. **Note** You can configure username capture from either `username_header` or `JWT` object, but not both.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
JWT `location` `username` `client id`	When the parameter values of `JWT` object are set, ASE decodes the JWT to extract the user information. location The place of occurrence of JWT in an API request. The supported values are: • `qs:<key name>` • `h:<custom header name>` • `h:authorization:bearer` • `h:authorization:mac` • `h:cookie:<cookie key>` username The JWT claim to extract the username. clientid The JWT claim to extract the client-id For more information, see Extract user information from JWT in inline mode. **Note** You can configure username capture from either `JWT` object or `username_header` , but not both.

Example Key	Value
X-Host	`%{HOST}` - If your backend server requires the value of header to contain host name or IP address of the server, use `%{HOST}` in value. During health check, ASE dynamically replaces header values containing %{HOST} with host name or IP address of the server. In the sample API JSON, ASE will dynamically replace %{HOST} with IP address (127.0.0.1) configured in the `servers` section. `"servers": [ { "host": "127.0.0.1", "port": 8080, "server_spike_threshold": "0/second", "server_connection_quota": 0 } ],`
X-Custom-Header	Your custom header value. All the custom health check headers configured are sent to all the backend API servers.

Example

The following is a sample JSON file for a REST API:

Copyright © 2025 Ping Identity Corporation

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/rest",
    "hostname": "*",
    "cookie": "",
    "cookie_idle_timeout": "200m",
    "logout_api_enabled": false,
    "cookie_persistence_enabled": false,
    "oauth2_access_token": false,
    "is_token_mandatory": false,
    "apikey_qs": "",
    "apikey_header": "",
    "login_url": "",
    "enable_blocking": true,
    "api_mapping": {
      "internal_url": ""
    },
    "api_pattern_enforcement": {
      "protocol_allowed": "",
      "http_redirect": {
        "response_code": "",
        "response_def": "",
        "https_url": ""
      },
      "methods_allowed": [],
      "content_type_allowed": "",
      "error_code": "401",
      "error_def": "Unauthorized",
      "error_message_body": "401 Unauthorized"
    },
    "flow_control": {
      "client_spike_threshold": "0/second",
      "server_connection_queueing": false
    },
    "api_memory_size": "128mb",
    "health_check": false,
    "health_check_interval": 60,
    "health_retry_count": 4,
    "health_url": "/health",
    "health_check_headers": {},
    "server_ssl": false,
    "servers": [
      {
        "host": "127.0.0.1",
        "port": 8080,
        "server_spike_threshold": "0/second",
        "server_connection_quota": 0
      },
      {
        "host": "127.0.0.1",
        "port": 8081,
        "server_spike_threshold": "0/second",
        "server_connection_quota": 0
      }
    ],
    "decoy_config": {
      "decoy_enabled": false,
      "response_code": 200,
      "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
        "response_message": "",
        "decoy_subpaths": []
      },
      "username_header": "x-username-header",
      "jwt": {
        "location": "h:authorization:bearer",
        "username": "username",
        "clientid": "client_id"
      }
    }
  }
```

Adding configured API

Adding a configured API JSON file to ASE
Before you begin

Configure an API JSON file. For more information, see API JSON files configuration.

About this task

To add an API JSON file to ASE to activate ASE processing:

Steps

1. To add an API, run the following command-line interface (CLI) command:

   ```
   /opt/pingidentity/ase/bin/cli.sh –u admin -p admin add_api {file_path/api_name}
   ```

2. Repeat step 2 for each API you want to add.

Copyright © 2025 Ping Identity Corporation

Updating configured API

Updating a configured API
About this task

After activation, to update an API JSON definition in real time:

Steps

1. Edit the API JSON file located in the `/config/api` directory and make the desired changes.
2. Save the edited API JSON file and run the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh –u admin -p admin update_api  <api_name>
```

Result:

```
/opt/pingidentity/ase/bin/cli.sh –u admin -p admin update_api shop
api shop updated successfully
```

API routing

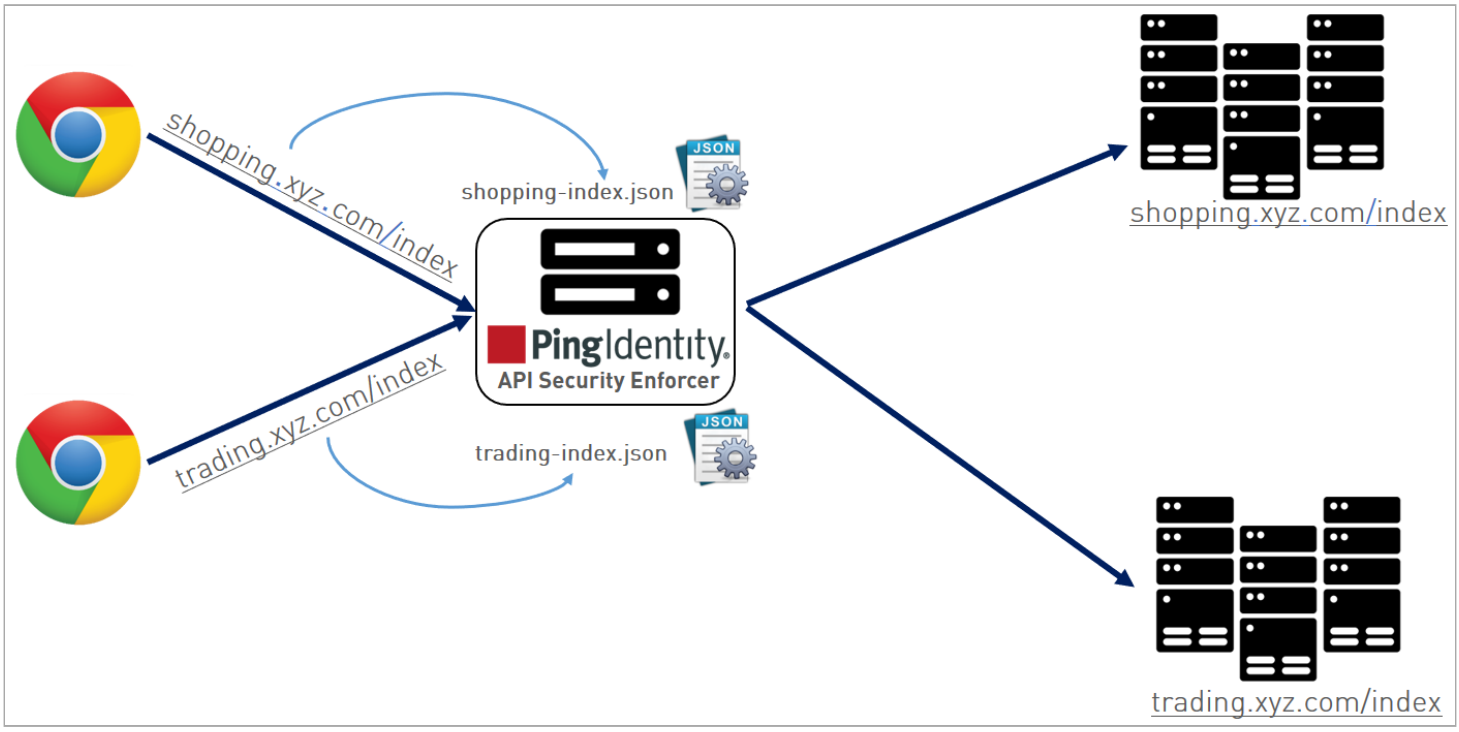
API Security Enforcer (ASE) uses a combination of header hostname and URL suffix to route incoming API requests to the correct backend server.

The following sections show scenarios for routing based on server and application programming interface (API) name:

Multiple host names with same API name

ASE supports configuring more than one hostname on one ASE node or cluster. It routes the incoming traffic based on the host name and the API configured in the JavaScript Object Notation (JSON) file. For example, traffic to two hosts named shopping.xyz.com and trading.xyz.com is routed based on the configurations in the respective API JSON file.

Copyright © 2025 Ping Identity Corporation



For incoming API requests, ASE first checks for the host name in the JSON file. If the host name is configured, then it checks for the API name. If both host and API name are defined, then the incoming API request is routed to one of the configured servers.

In the above example, ASE checks whether shopping.xyz.com is configured in the JSON file (`shopping.json`). It then checks for the API, `/index` . If it finds both to be present, then it routes the traffic to one of the defined backend servers. The following is a snippet from a sample JSON file which shows the values that should be configured for `shopping.json` :

```
"api_metadata": {
 "protocol": "https",
 "url": "/index,
 "hostname": "shopping.xyz.com",
 "cookie": "JSESSIONID",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": true,
 "cookie_persistence_enabled": false,
```

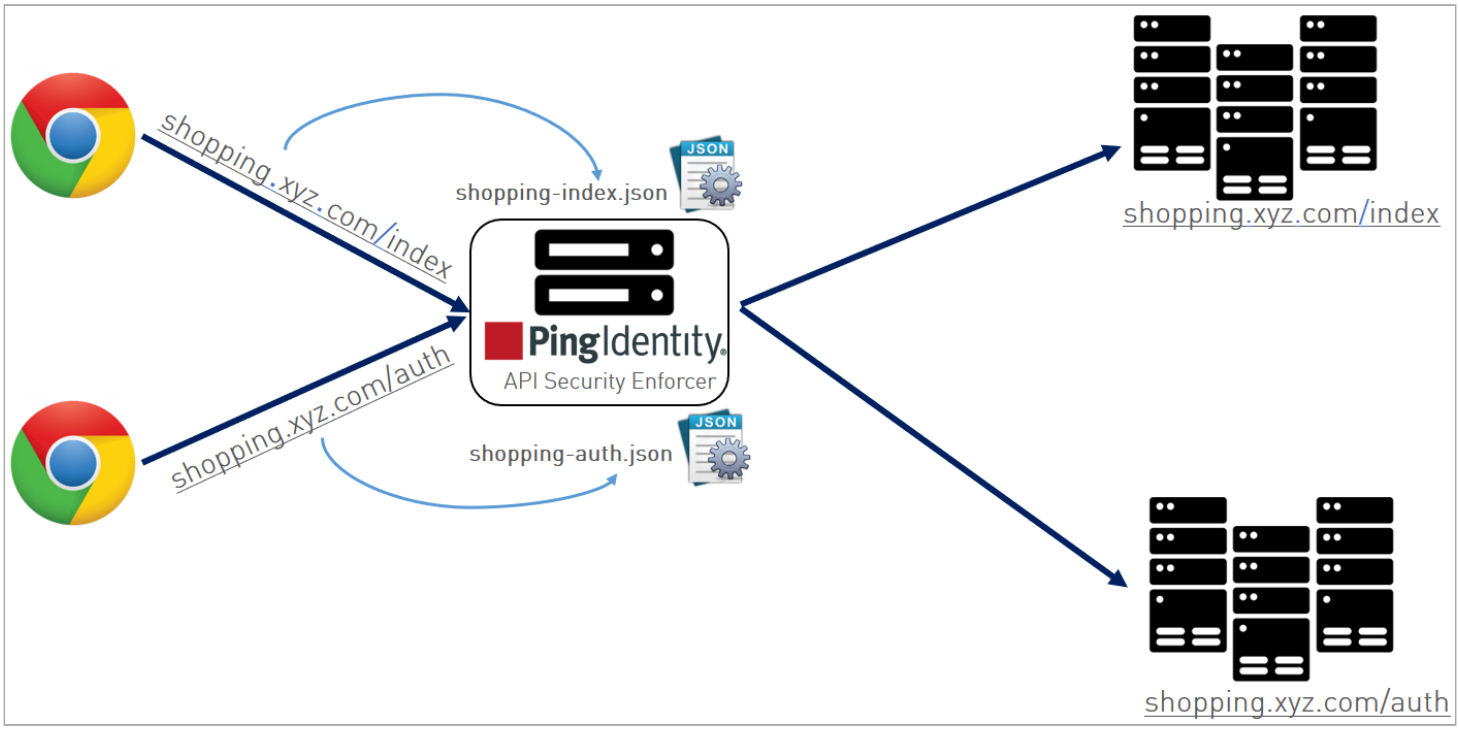
For each API, configure a separate JSON file.

Single host name with different API names

ASE supports configuring the same hostname with different API names. For example, hostname shopping.xyz.com has two different APIs, `/index` and `/auth` . Traffic to each API is routed using the API specific JSON file: `shopping-index.json` or `shopping-auth.json` .

In the following illustration, any requests for shopping.xyz.com/index are routed by ASE to a server configured in `shopping-index.json` . In this case, shopping-index.json file parameters must match for both the hostname and API. Similarly, requests to shopping.xyz.com/auth, are routed by ASE to a server configured in `shopping-auth.json` .

Copyright © 2025 Ping Identity Corporation



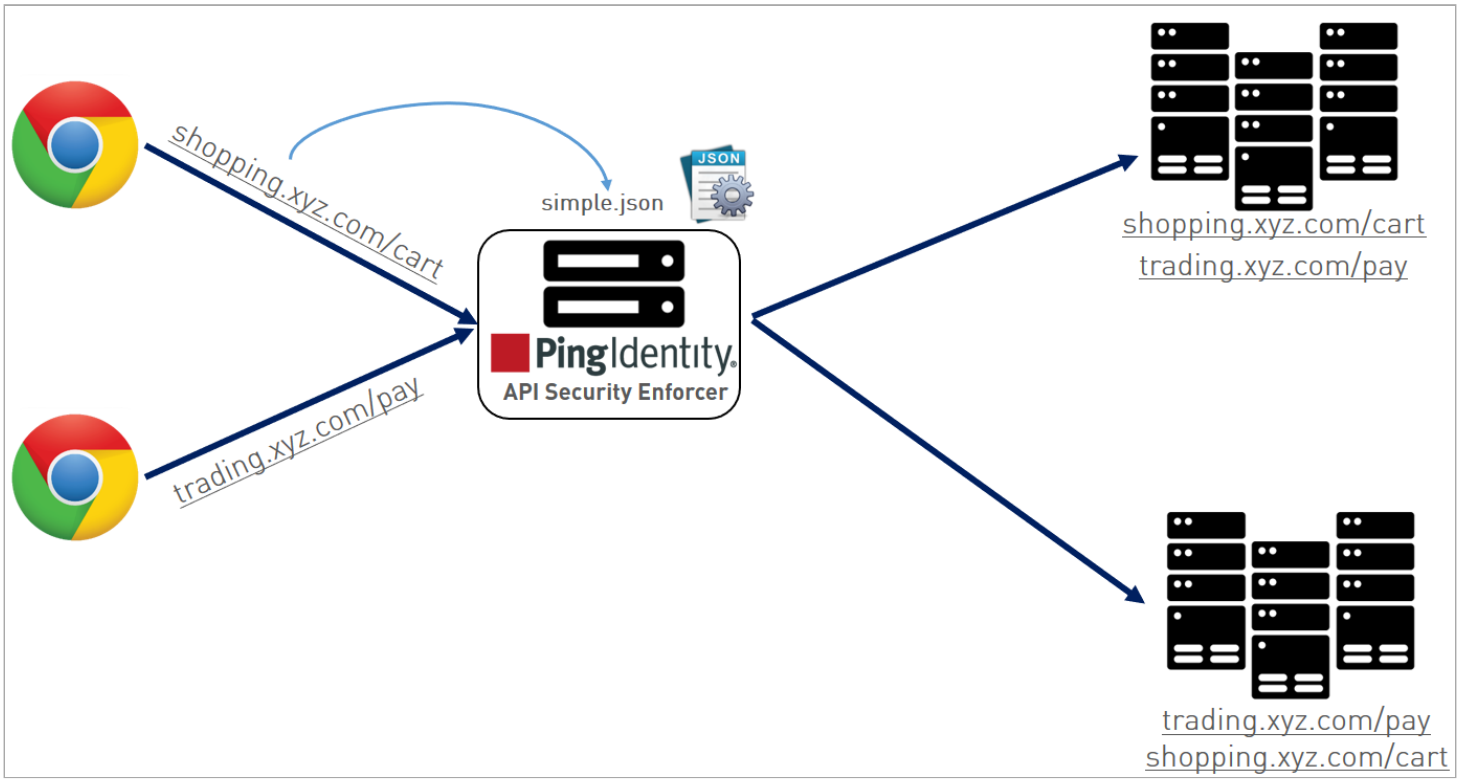
Wildcard host name and API name

ASE can also be used as a simple load balancer to route traffic for legacy web applications. The load balancing technique used for server load balancing is based on protocol and cookie information. To configure ASE as a simple load balancer, set the following parameters in a JSON file:

```
“hostname”: “*”,
"url": "/",
```

When hostname `“*”` and `url` `“/”` are configured in a JSON file, any request that does not match a specific hostname and `url` defined in another JSON file uses the destination servers specified in this file to route the traffic.

Copyright © 2025 Ping Identity Corporation



In the above illustration, hostname is configured as `"*"` and `url` as `"/"`. ASE does not differentiate between hostname and API name. It simply balances traffic across all backend servers.

For all scenarios, when connections are being routed to a backend server which goes down, ASE dynamically redirects the connections to a live server in the pool.

Enabling and disabling real-time API cybersecurity

API Security Enforcer (ASE) provides real-time API cybersecurity to stop hackers. Violations are immediately blocked, and attack information is sent to the API Behavioral Security (ABS) engine.

About this task

> **Note**
>
> Real-time API cybersecurity is activated only when ASE firewall is enabled. For more information on enabling the ASE firewall, see CLI for inline ASE.

Steps

- To enable API cybersecurity:

  1. Run the `enable_firewall` command in the command-line interface (CLI):

     Example:

     ```
     /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
     Firewall is now enabled
     ```


Copyright © 2025 Ping Identity Corporation

2. To verify that cybersecurity is enabled, run the `status` command:

   Example:

```
/opt/pingidentity/ase/bin/cli.sh status
Ping Identity Inc., API Security Enforcer
status : started
http/ws : port 80
https/wss : port 443
firewall : enabled
abs : disabled
abs attack : disabled
audit : enabled
ase detected attack : disabled
attack list memory : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

- To disable API cybersecurity:

  1. Run the `disable_firewall` command:

     Example:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_firewall
Firewall is now disabled
```

  2. To verify that cybersecurity is enabled, run the `status` command:

     Example:

```
/opt/pingidentity/ase/bin/cli.sh status
Ping Identity Inc., API Security Enforcer
status : started
http/ws : port 80
https/wss : port 443
firewall : disabled
abs : disabled
abs attack : disabled
audit : enabled
ase detected attack : disabled
attack list memory : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

ASE attack detection

API Security Enforcer (ASE) supports the following real-time ASE attack detection and blocking. For more information on enabling and disabling ASE-detected attacks, see Enabling and disabling ASE attack detection.

- API pattern enforcement – validate traffic to ensure it is consistent with the application programming interface (API) definition
- API deception – blocks hackers probing a decoy API. For more information, see API deception environment in inline mode.

Copyright © 2025 Ping Identity Corporation

Pattern enforcement configuration

After enabling API cybersecurity, configure API pattern enforcement to block API traffic that does not match the permitted criteria in the following categories:

- Protocol (HTTP, HTTPS, WS, WSS) – only allow the defined protocols
- Method (GET, POST, PUT, DELETE, HEAD) – only allow the specified methods
- Content Type – only allow the defined content type, not enforced if an empty string is entered
- HTTPS Only – only allow HTTPS traffic

ASE blocks attacks based on parameters configured in the API JavaScript Object Notation (JSON) file. If a client request includes values not configured in the API JSON, ASE blocks the connection in real time. When the connection is blocked, the OAuth2 token, cookie, or IP address is blocked from accessing any APIs.

The following API JSON file snippet shows an example of pattern enforcement parameters:

```
"api_pattern_enforcement": {
 "protocol_allowed": "https",
 "http_redirect": {
 "response_code": 301,
 "response_def": "Moved Permanently",
 "https_url": "https://shopping.xyz.com/login/"
 },
 "methods_allowed": [
 "GET",
 "POST"
 ],
 "content_type_allowed": "application/json",
 "error_code": 401,
 "error_def": "Unauthorized",
 "error_message_body": " Error: Unauthorized"
 },
```

The above example sets up the following enforcement:

- Only HTTPS traffic is allowed access to the API. If an HTTP request is sent, it will be redirected to the `https_url` defined in the `http_redirect` section.
- Only GET and POST methods are allowed. PUT, DELETE, and HEAD will be blocked.
- Only application/JSON content type is allowed. Other content types are blocked.

If a request satisfies all three parameters (protocol, method, and content type), ASE will send the request to the backend API server for processing. Otherwise, ASE sends an error code using the following API JSON parameters:

Parameter	Error code
`Error_code`	Error code, for example: “401”

Copyright © 2025 Ping Identity Corporation

Parameter	Error code
`error_def`	Error definition, for example: “Unauthorized”
`error_message_body`	Error message content, for example: “Error: Unauthorized”

If an empty string is specified for `content_type_allowed`, ASE does not enforce content type for the incoming traffic.

```
"content_type_allowed": ""
```

> **Note**
>
> When API security is enabled, the `protocol_allowed` parameter takes precedence over the `protocol` parameter in the beginning of the API JSON file.

Detection of attacks for pattern enforcement violation

The following is a snippet of access log file showing what is logged when a connection is blocked based on any pattern enforcement violation.

> **Note**
>
> Make sure that ASE-detected attacks are enabled.

The following example shows a method violation for an OAuth2 token:

```
[Fri Aug 10 15:59:12:435 2018] [thread:14164] [info] [connectionid:1681692777] [seq:1] [connectinfo:
100.100.1.5:36839] [type:request] [api_id:shop] PATCH /shopapi/categories/list HTTP/1.1
User-Agent: curl/7.35.0
Accept: /
Host: app
Content-Type: application/text
Cookie: JSESSIONID=ebcookie
Authorization: Bearer OauthTokenusemethoid12345
[Fri Aug 10 15:59:12:435 2018] [thread:14164] [info] [connectionid:1681692777] [seq:1] [connectinfo:
100.100.1.5:36839] [type:connection_drop]  [enforcement:method]  [api_id:shop] PATCH /shopapi/categories/
list HTTP/1.1
User-Agent: curl/7.35.0
Accept: /
Host: app
Content-Type: application/text
Cookie: JSESSIONID=ebcookie
Authorization: Bearer OauthTokenusemethoid12345
```

Violations logged in the ASE access log files are sent to API Behavioral Security (ABS) AI Engine for further analysis and reporting.

Copyright © 2025 Ping Identity Corporation

API name mapping – hide internal URLs

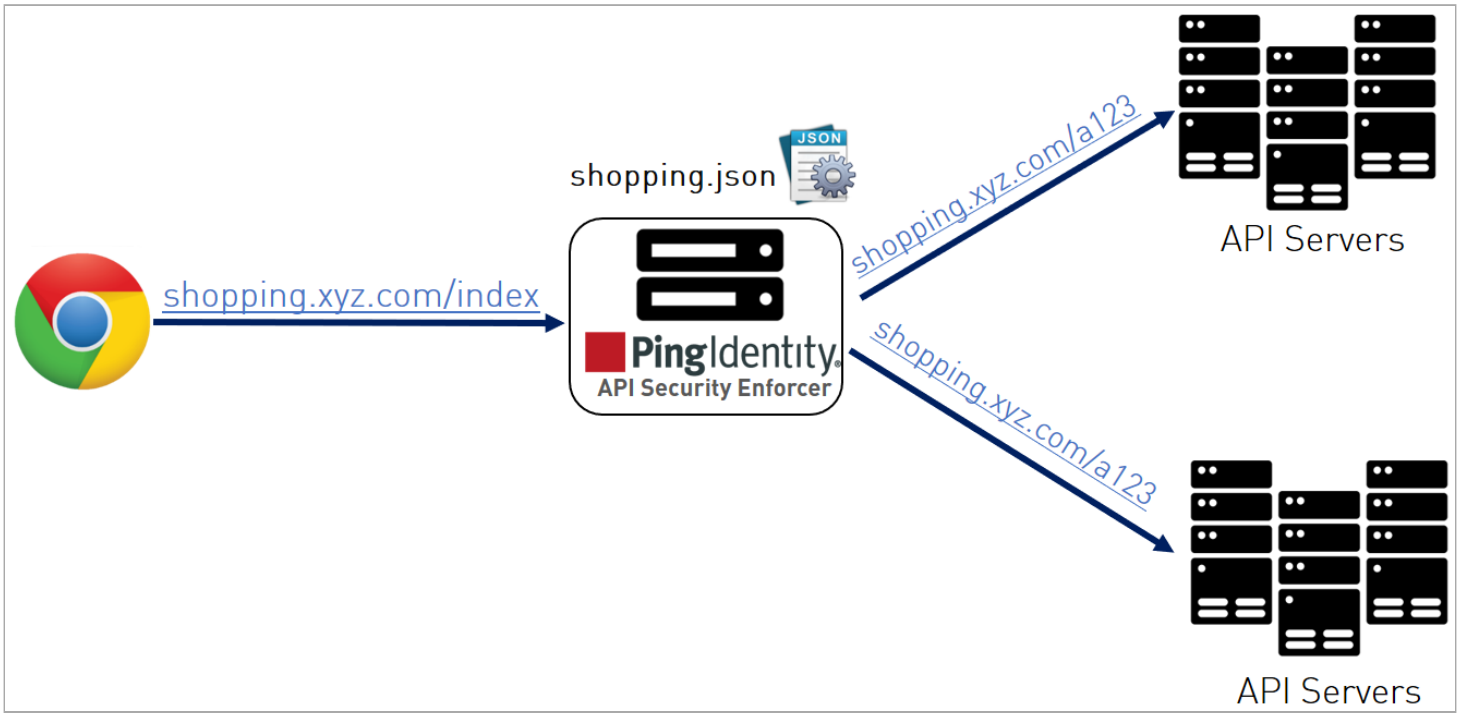
After enabling application programming interface (API) cybersecurity, API name mapping can be configured to protect API servers by hiding internal Uniform Resource Locator (URL)s from the outside world. Internal URLs may also be modified without updating entries in the public DNS server.

For example, the following JavaScript Object Notation (JSON) snippet from an API JSON file maps an external URL ("/index") for shopping.xyz.com to an internal URL ("/a123").

```
"api_metadata": {
 "protocol": "http",
 "url": "/index",
 "hostname": "127.0.0.1",
 "cookie": "JSESSIONID",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": true,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "cookie_persistence_enabled": true,
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "login_url": "/index/login",
 "api_mapping": {
 "internal_url": "/a123"
 },
```

The following diagram illustrates the data flow from the client to the backend server through the API Security Enforcer (ASE):

Copyright © 2025 Ping Identity Corporation



Capture client identifiers in inline mode

API Security Enforcer (ASE) identifies attackers for HTTP(s) and WS(s) protocols using five client identifiers:

- **OAuth2 token**
- **Cookie**
- **IP address**
- **API keys**
- **Username**

> **Note**
>
> Username is not configured in the api_metadata object of application programming interface (API) JavaScript Object Notation (JSON). However, ASE supports the extraction of usernames coming in a JSON Web Token (JWT) or custom headers. For more information, see Extract user information from JWT in inline mode and Extract username from custom header in inline mode. For usernames that are not part of either JWTs or custom headers, API Behavioral Security (ABS) AI Engine identifies them based on metadata logged in ASE's access logs.

The following sections describe how to configure ASE to capture OAuth2 Tokens and API keys.

Copyright © 2025 Ping Identity Corporation

Configure ASE support for OAuth2 tokens

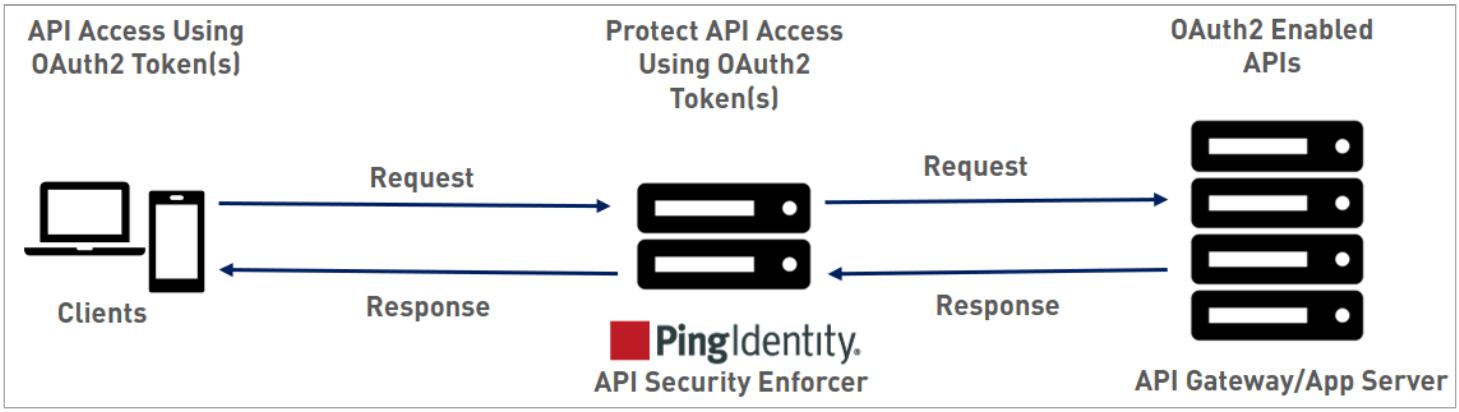
ASE supports capturing and blocking of OAuth2 tokens. To enable OAuth2 token capture, set the value of `oauth2_access_token` to `true` in the API JSON file. Here is a snippet of an API JSON file with OAuth2 Token capture activated. To disable, change the value to `false`.

```
"api_metadata": {
        "protocol": "http",
         "url": "/",
         "hostname": "*",
         "cookie": "",
         "cookie_idle_timeout": "200m",
         "logout_api_enabled": false,
         "cookie_persistence_enabled": true,
         "oauth2_access_token": true,
         "is_token_mandatory": false,
         "apikey_qs": "",
         "apikey_header": "",
         "login_url": "",
         "enable_blocking": true,
         "api_mapping": {
            "internal_url": ""
                              },
```

When `enable_blocking` is `true`, ASE checks the token against the list of tokens in the allow list and deny list. If the token is in the deny list, the client using the token is immediately blocked. Further, when `is_token_mandatory` is set to `true`, and the incoming request has a missing token, ASE adds the IP address of the client to deny list and blocks the request.

> **Important**
>
> For ASE to check and block the client, `enable_firewall` and `enable_ase_detected_attack` must be set to `true` in Inline ASE configuration using the ase.conf file.

When pattern enforcement violations are detected on an API configured to support tokens, the attacking client token is added to the deny list in real-time, recorded in the ASE access log, and sent to ABS for further analytics. The following diagram shows the traffic flow in an OAuth2 environment:



Copyright © 2025 Ping Identity Corporation

Configure ASE support for API keys

ASE supports capturing and blocking of API keys. Depending on the API setup, the API key can be captured from the query string or API header. Each API JSON file can be configured with either the query string (`apikey_qs`) or API header (`apikey_header`) parameter.

Here is a snippet of an API JSON file showing API Key being configured to capture the API Key from the Query String (`apikey_qs`).

```
"api_metadata": {
                                "protocol": "http",
                                "url": "/",
                                "hostname": "*",
                                "cookie": "",
                                "cookie_idle_timeout": "200m",
                                "logout_api_enabled": false,
                                "cookie_persistence_enabled": true,
                                "oauth2_access_token": true,
                                "is_token_mandatory": false,
                                "apikey_qs": "key_1.4",
                                "apikey_header": "",
                                "login_url": "",
                                "enable_blocking": true,
                                "api_mapping": {
                                "internal_url": ""
                                },
```

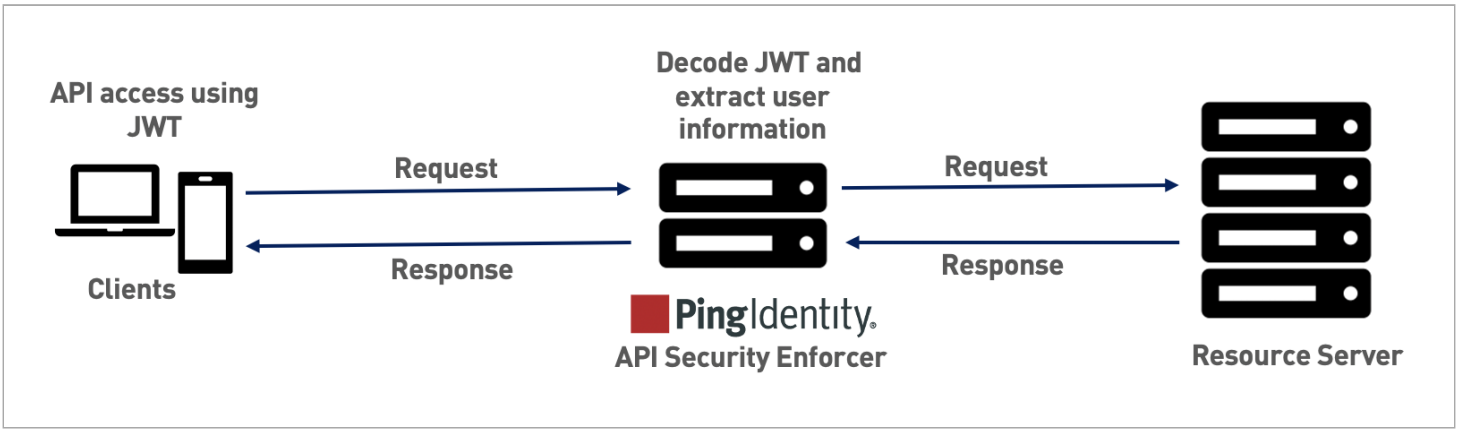
When an API Key is included in the API JSON file, ASE supports blocking of API keys which are manually added to the deny list.

Extract user information from JWT in inline mode

API Security Enforcer (ASE) supports the decoding of transparent JSON Web Token (JWT)s received as part of application programming interface (API) requests. It extracts the user information from the JWT and logs it in the ASE access logs. The API Behavioral Security (ABS) AI engine analyzes these access logs to detect attacks and anomalies.

The following diagram shows the traffic flow when ASE is in inline mode.

Copyright © 2025 Ping Identity Corporation



A JWT consists of three parts, header, payload, and signature, concatenated with periods (.). The following image shows a sample JWT structure.

ASE decodes the payload to extract user information from a JWT. ASE can decode JWTs received as part of request headers or query strings. In inline mode, ASE supports `Bearer` and `MAC` schemes in the `Authorization` header.

> **Note**
>
> ASE decodes the JWTs and extracts the user information. It does not validate JWTs.

ASE supports a list of usernames in JWT. When the `username` claim in the payload is an array with multiple elements, ASE extracts the first element of the array. The elements in the array can be strings or numbers, and the array should be a valid JSON array.

```
{
  "username": ["user1", "user2", "user3", "user4"],
  "clientid": "client1",
  "location": "Bearer"
}
```

> **Note**
>
> ASE supports arrays only for username claims in the payload. It does not support arrays in clientid or location claims.

Copyright © 2025 Ping Identity Corporation

When ASE is deployed in inline mode, it decodes the JWTs only when the `username` and `location` values are configured in an API JSON file for the API.

> **Note**
>
> If the JWT decoding fails, the API request is not blocked. ASE logs the metadata in the access logs.

The API JSON file

The behavior and properties of your API are defined in an API JSON file in ASE. To enable username capture, set the values for the parameters defined in the JWT object of the API JSON file as per your API setup. For more information, see Defining an API using API JSON configuration file in inline mode.

The following is an example snippet of an API JSON file.

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/rest",
    "hostname": "*",
    "cookie": "",
    "cookie_idle_timeout": "200m",
    "logout_api_enabled": false,
    "cookie_persistence_enabled": false,
    "oauth2_access_token": true,
    "apikey_qs": "",
    "apikey_header": "",
    "login_url": "",
    "enable_blocking": true,
    "api_mapping": {
      "internal_url": ""
    },

    "username_header": "",
    "jwt": {
      "location": "h:authorization:bearer",
      "username": "username",
      "clientid": "client_id"
    }
  }
}
```

> **Note**
>
> The values assigned to `username` and `clientid` must be different.

The following table describes the parameters in the JWT object of API JSON file.

Copyright © 2025 Ping Identity Corporation

`location`	The JWT location in an API request. Configure the parameter with a value applicable to your API.
	The supported values for the `location` parameter are:
	
	`qs: <key name>`
	Set the location parameter with this value when JWT occurs as part of a query string and substitute the *<key name>* with the query string parameter. For example, `"location": "qs:access_token"`.
	
	`https://server.example.com/resource?`
	`access_token=mF_9.B5f-4.1JqM&p=q`
	
	`h: <custom header name>`
	Set the location parameter with this value when JWT is part of a custom header and substitute the *<custom header name>* with custom header. For example, `"location": "h:X-jwt-header"`.
	
	`X-jwt-header:`
	`eyJhbGcUzI1NiI.eyJzDkwIG4gRG9xpZWQiOjwMjJ9.DWw5PDZEl-g`
	
	`h:Authorization:bearer`
	Set the location parameter with this value when JWT is part of Authorization header, with bearer scheme. For example, `"location": "h:Authorization:Bearer"`.
	
	`Authorization: Bearer`
	`eyJhbGIUzIiI.eyJzdiIxG4gRG9lIiwiZiOjJ9.DWPwNDZEl-g`
	
	`h:Authorization:MAC`
	Set the location parameter with this value when JWT is part of Authorization header, with MAC scheme. For example, `"location": "h:Authorization:MAC"`.
	
	`Authorization: MAC id="eyJhbGcI1NiI",`
	`nonce="272095:dp63hm5s",`
	`mac="PNPQW4mg43cjQfEpUs3QWub4o6xE="`
	
	`h:cookie:<cookie key>`
	Set the location parameter with this value when JWT occurs as part of a cookie and substitute the *<cookie key>* with the cookie name. For example, `"location": "h:cookie: access_token"`.
	
	`Cookie:`
	`access_token=eyJhbGiIsI.eyJpc3MiOiJodHRwczotcGxlL.mFrs3ZodqKP4F1cB`

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`username`	The JWT claim to extract the username.
`clientid`	The JWT claim to extract the client-id.

When `enable_blocking` is set to `true`, ASE checks the username against the list of usernames in the allow list and deny list. If the username is in the deny list, the client using the username is blocked.

> **Note**
>
> ASE also supports extracting username from a custom HTTP header. However, you can configure username capture from either custom header or JWT, but not both. For more information, see Extract username from custom header in inline mode.

The API discovery processThe ABS AI Engine processes the ASE access logs and discovers new and unknown APIs in your environment. A root API JSON is defined in ASE to enable API discovery by ABS. For more information on API discovery, see API discovery and configuration. If the root API JSON has a JWT object configured with values set for all the keys, then the APIs discovered by the ABS will have the JWT object.

The following table explains the behavior of ASE when the API JSON has an incomplete JWT object and describes its impact on the APIs discovered by ABS in your environment.

Scenarios	Behavior of ASE	Impact on API discovery
A JWT object is not configured in API JSON.	ASE processes the API JSON file.	A JWT object gets added to the discovered APIs with all the keys but empty values. For example. `"jwt": { "username": "", "clientid": "", "location": "" }`
A JWT object is configured in API JSON file but with no keys. For example. `“jwt”:{}`	ASE does not process the API JSON file.	The API is not discovered.

Copyright © 2025 Ping Identity Corporation

Scenarios	Behavior of ASE	Impact on API discovery
A JWT object is configured with all the keys present but with no values set. For example. `"jwt": { "username": "", "clientid": "", "location": "" }`	ASE processes the API JSON file.	A JWT object gets added to the discovered APIs with all the keys but empty values. For example. `"jwt": { "username": "", "clientid": "", "location": "" }`
When a JWT object is configured, but not all keys are set. For example. `"jwt": { "username": "", "location": "" }`	ASE does not process the API JSON file.	The API is not discovered.

Note

The API JSON file shipped with ASE is compatible with earlier versions of API JSON files. ASE automatically adds an empty JWT object to the API JSON file to maintain compatibility.

Extract username from custom header in inline mode

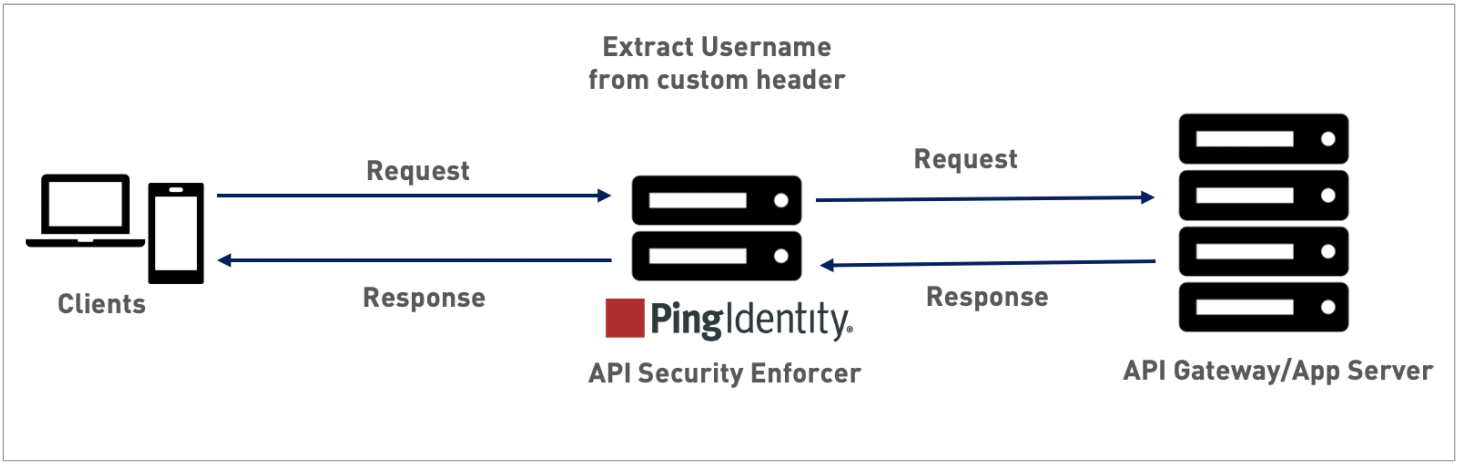
This topic discusses the extraction of a username from a custom header when API Security Enforcer (ASE) is in inline mode.

ASE supports capturing usernames from custom headers in a request. It extracts the username and logs it in ASE access logs. ASE sends these access log files to API Behavioral Security (ABS) AI Engine to detect attacks. The following is an example of username information logged in the ASE access log:

Copyright © 2025 Ping Identity Corporation

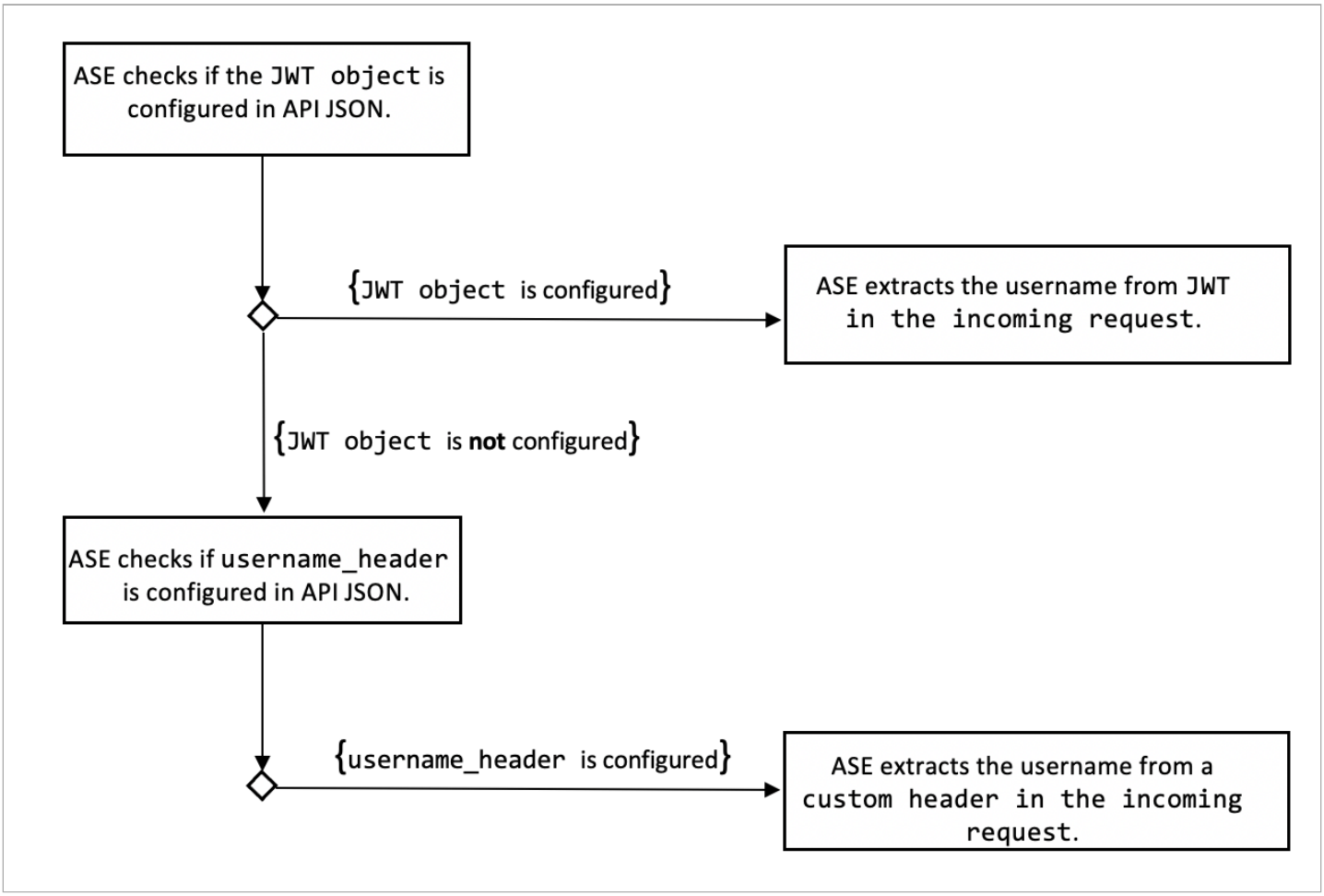
```
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [connectinfo:127.0.0.0:80]
[type:connection] connection received
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:request] [api_id:api1] GET /abcd HTTP/1.1
x-username-header: 12n4uf9ckls

host: http://pi-api-mngmnt.azr-api.net/
accept: /
content-type: text/plain;charset=UTF-8

[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:backend_info] [backend_type:nonssl] [0] [api_id:api1] [hostname:not available] backend
selected
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:req_payload] [api_id:api1] [size:0]
[Tue Dec 15 09:13:45:044 2020] [thread:999] [info] [connectionid:1801979802] [seq:1] [connectinfo:
127.0.0.0:80] [type:user_info] [api_id:api1] username: 12n4uf9ckls
```

The following diagram shows the traffic flow when ASE is in inline mode.



When deployed in inline mode, ASE extracts the username from either JSON Web Token (JWT) or a custom header. It checks the configuration of application programming interface (API) JavaScript Object Notation (JSON) file. It first checks the `JWT` object. If it is configured, then ASE will capture the username from a JWT in the incoming request. Otherwise, ASE checks the `username_header` parameter in API JSON. If it is set, ASE extracts the username from the custom header that comes as part of an incoming request. For more information, see the Configure API JSON section below.

Copyright © 2025 Ping Identity Corporation



Important

ASE supports extracting username from either JWTs or a custom headers. You can configure API JSON to capture username from either custom header or JWT, but not both for a given API. For more information on extracting usernames from JWTs, see Extract user information from JWT in inline mode.

API JSON configuration in inline mode

The behavior and properties of your API are defined in an API JSON file in the ASE. To enable username capture from a custom header, set the value of the `username_header` parameter to the custom header name containing the username. The following is an example of an API JSON file.

Copyright © 2025 Ping Identity Corporation

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/rest",
    "hostname": "*",
    "cookie": "",
    "cookie_idle_timeout": "200m",
    "logout_api_enabled": false,
    "cookie_persistence_enabled": false,
    "oauth2_access_token": false,
    "apikey_qs": "",
    "apikey_header": "",
    "login_url": "",
    "enable_blocking": true,

    "api_mapping": {
      "internal_url": ""
    },
    "api_pattern_enforcement": {
      "protocol_allowed": "",
      "http_redirect": {
        "response_code": "",
        "response_def": "",
        "https_url": ""
      },
      "methods_allowed": [],
      "content_type_allowed": "",
      "error_code": "401",
      "error_def": "Unauthorized",
      "error_message_body": "401 Unauthorized"
    },
    "flow_control": {
      "client_spike_threshold": "0/second",
      "server_connection_queueing": false
    },
    "api_memory_size": "128mb",
    "health_check": false,
    "health_check_interval": 60,
    "health_retry_count": 4,
    "health_url": "/health",
    "health_check_headers": {},
    "server_ssl": false,
    "servers": [
      {
        "host": "127.0.0.1",
        "port": 8080,
        "server_spike_threshold": "0/second",
        "server_connection_quota": 0
      },
      {
        "host": "127.0.0.1",
        "port": 8081,
        "server_spike_threshold": "0/second",
        "server_connection_quota": 0
      }
    ],
    "decoy_config": {
      "decoy_enabled": false,
      "response_code": 200,
      "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
        "response_message": "",
        "decoy_subpaths": []
      },
      "username_header": "x-username-header",
      "jwt": {
        "location": "",
        "username": "",
        "clientid": ""
      }
    }
  }
```

For more information, see Defining an API using API JSON configuration file in inline mode.

You can optionally block a client. When `enable_blocking` is set to `true`, ASE checks the username against the list of usernames in the allow list and deny list. If the username is in the deny list, the client using the username is blocked.

> **Note**
>
> The API JSON file shipped with ASE is compatible with earlier versions of API JSON files. ASE automatically adds an optional `username_header` parameter to the API JSON file to maintain compatibility.

Managing allow lists and deny lists

The API Security Enforcer (ASE) maintains both allow lists and deny lists.

Allow list

List of safe IP addresses, cookies, OAuth2 tokens, API keys, or usernames that are not blocked by ASE.The list is manually generated by adding the client identifiers using command-line interface (CLI) commands.

Deny list

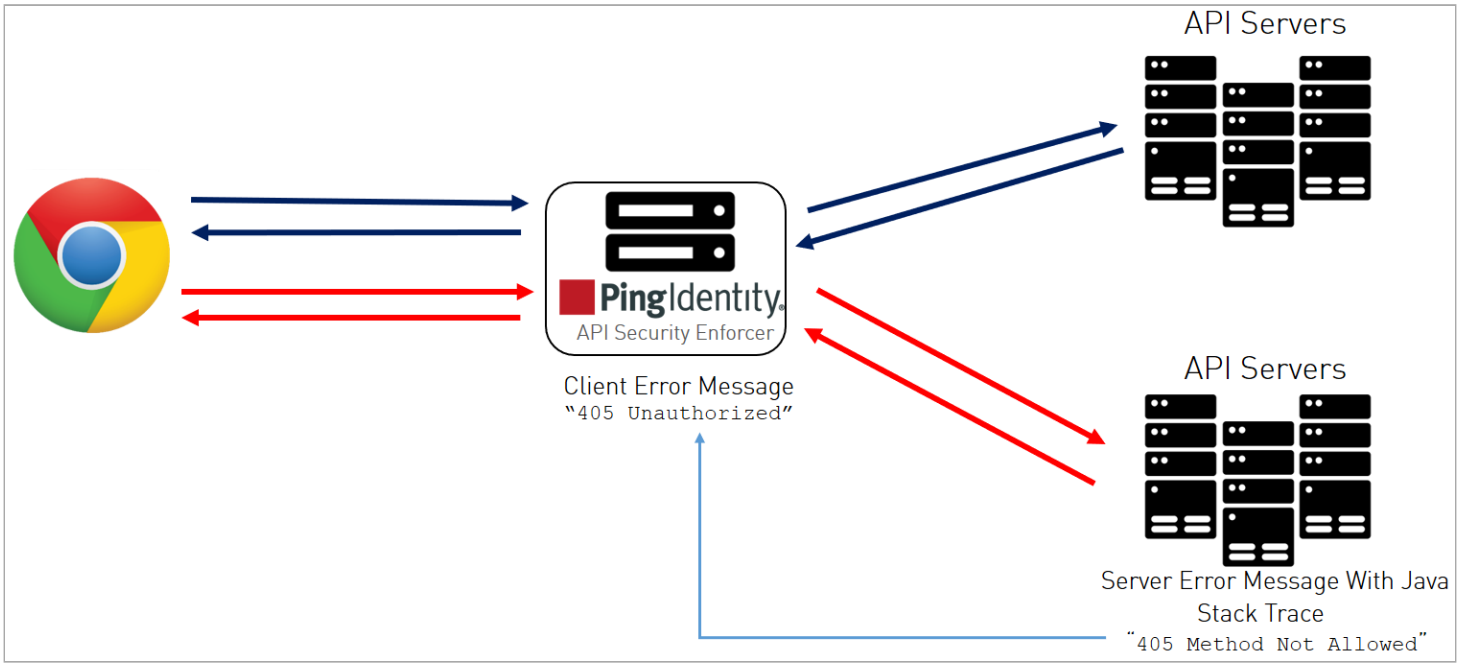
List of bad IP addresses, cookies, OAuth2 tokens, API keys, or usernames that are always blocked by ASE.The list consists of entries from one or more of the following sources:

- API Behavioral Security (ABS)-detected attacks, such as data exfiltration. ABS-detected attacks have a time-to-live (TTL) in minutes. The TTL is configured in ABS.
- ASE-detected attacks, such as invalid method or decoy API accessed.
- List of bad clients manually generated by CLI.

Map server error messages to custom error messages

Backend server error messages (for example, Java stack trace) can reveal internal information to hackers. API Security Enforcer (ASE) supports hiding the internal details and only sending a customized simple error message. The error message mappings are defined in `/config/server_error.json` file.

Copyright © 2025 Ping Identity Corporation



For each custom HTTP error code, specify all three parameters in `server_error.json`. For example, the snippet of `server_error.json` shows parameters for mapping error codes 500 and 503.

```
{
 "server_error": [
 {
 "error_code" : "500",
 "error_def" : "Internal Server Error",
 "msg_body" : "Contact Your Administrator"
 },
 {
 "error_code" : "503",
 "error_def" : "Service Unavailable",
 "msg_body" : "Service Temporarily Unavailable"
 }
 ]
}
```

In the above example, an ASE which receives an error 500 or 503 message from the application replaces the message with a custom name `error_def` and message `msg_body` as defined in the `server_error.json` file.

To send the original error message from the backend server, do not include the associated error code in the `server_error.json` file. An empty `server_error.json` file as shown below will not translate any backend error messages.

```
{
 "server_error": [
 ]
}
```

Copyright © 2025 Ping Identity Corporation

> **Important**
>
> ASE checks for the presence of the `server_error.json` file. If this file is not available, ASE will not start.

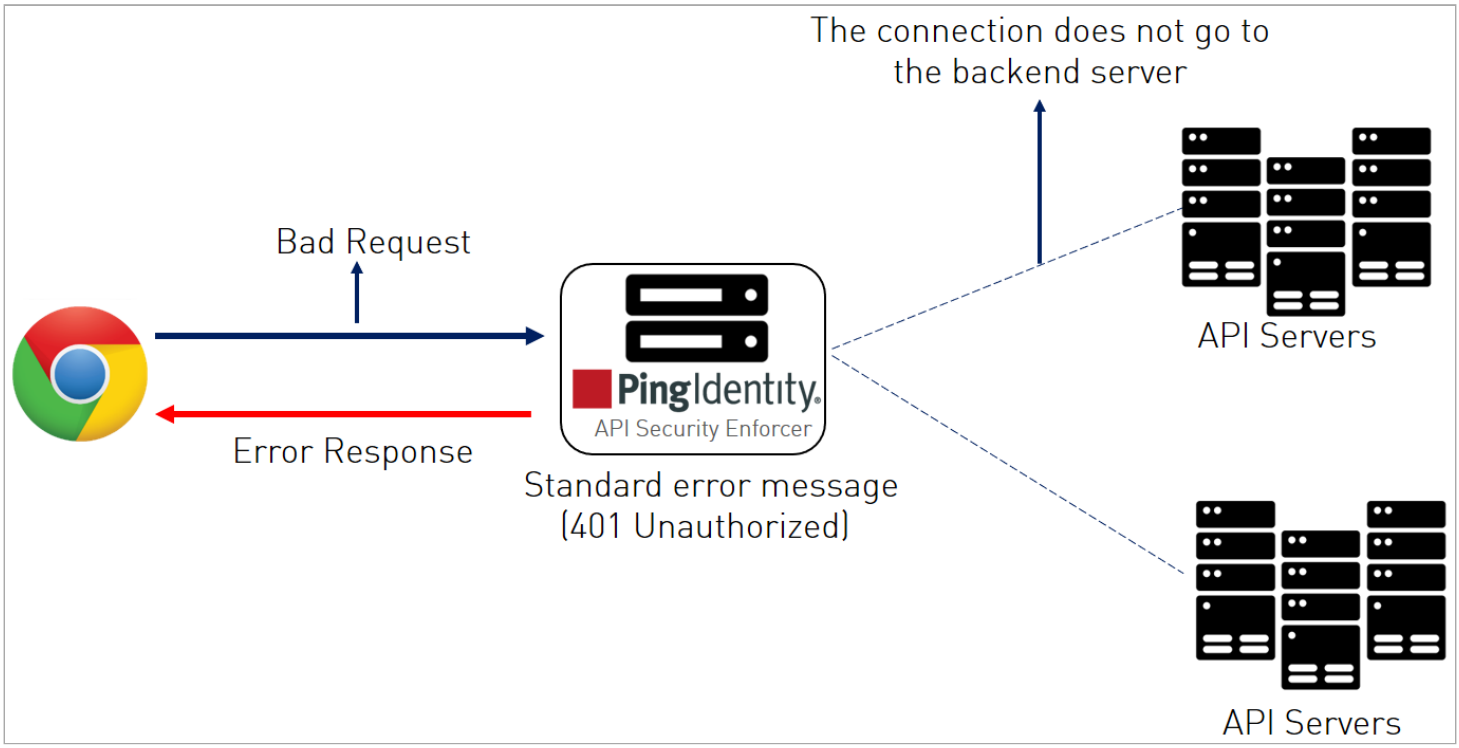
ASE-generated error messages for blocked requests

The API Security Enforcer (ASE) blocks certain requests based on application programming interface (API) Mapping or API Behavioral Security (ABS)-detected attacks. For these blocked requests, it sends a standard error message back to the client.

The following table describes the error messages.

Blocked Connection	HTTP Error Code	Error Definition	Message Body
Unknown API	503	Service Unavailable	Error: Unknown API
Unknown Hostname	503	Service Unavailable	Error: Unknown Hostname
Malformed Request	400	Bad Request	Error: Malformed Request
IP attack	403	Unauthorized	Error: Unauthorized
Cookie attack	403	Unauthorized	Error: Unauthorized
OAuth2 Token attack	403	Unauthorized	Error: Unauthorized
API Key attack	403	Unauthorized	Error: Unauthorized
Username attack	403	Unauthorized	Error: Unauthorized

Copyright © 2025 Ping Identity Corporation



Enabling per API blocking

You can configure the API Security Enforcer (ASE) to selectively block on a per application programming interface (API) basis by configuring an API JavaScript Object Notation (JSON) file parameter.

About this task

To enable per API blocking for each API:

Steps

- **Set the `enable_blocking` parameter to `true` in the API JSON file.**

  Example:

Copyright © 2025 Ping Identity Corporation

```
api_metadata": {
 "protocol": "http",
 "url": "/",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "enable_blocking": true,
 "login_url": "",
 "api_mapping": {
 "internal_url": ""
 },
```

> **Note**
>
> If per API blocking is disabled, API Behavioral Security (ABS) still detects attacks for that specific API. ASE does not block them. ASE continues to block attacks on other APIs with the `enable_blocking` parameter set to `true`.

API deception environment in inline mode

A decoy application programming interface (API) is configured in the API Security Enforcer (ASE) and requires no changes to backend servers. It appears as part of the API ecosystem and is used to detect the attack patterns of hackers.

When a hacker accesses a decoy API, ASE sends a predefined response (defined in `response_message` parameter in the API JavaScript Object Notation (JSON) file) to the client request and collects the request information as a footprint to analyze API ecosystem attacks. ASE does not forward decoy API request traffic to backend servers.

Decoy API traffic is separately logged in files named with the following format: `decoy_pid_<pid_number>yyyy-dd-mm-<log_file_rotation_time>` *(for example,* `decoy_pid_87872017-04-04_10-57.log`*). Decoy log files are rotated every 24 hours and stored in the* `opt/pingidentity/ase/logs` *directory.*

ASE provides the following decoy API types in inline mode:

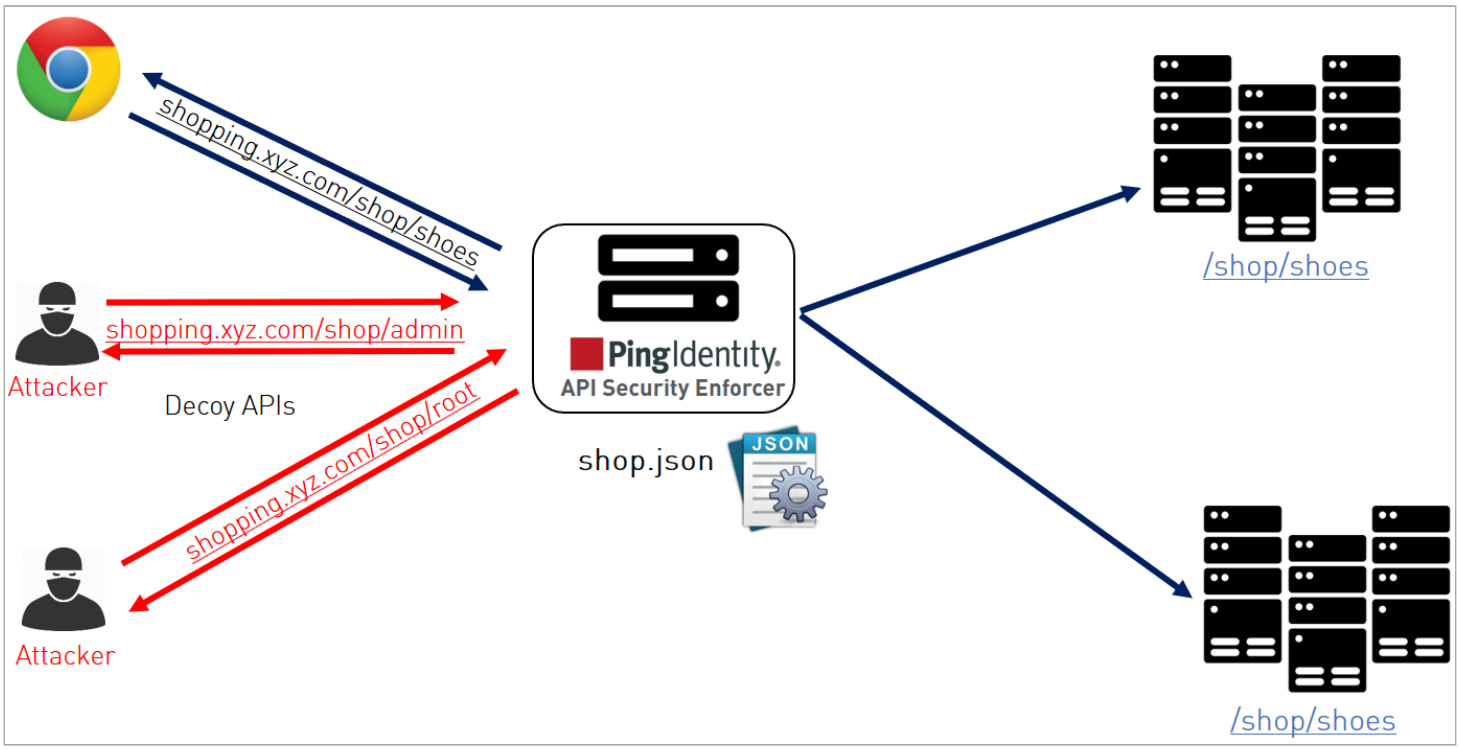
- *In-context decoy APIs*
- *Out-of-context decoy APIs*

In-context decoy API

In-context decoy application programming interface (API)s consist of decoy paths within existing APIs supporting legitimate traffic to backend servers.

Copyright © 2025 Ping Identity Corporation

Any traffic accessing a decoy path receives a preconfigured response. For example, in the `shopping` API, `/root` and `/admin` are decoy APIs ; and `/shoes` is a legitimate API path. Traffic accessing `/shoes` is redirected to the backend API server, while the traffic that accesses `/root` or `/admin` receives a preconfigured response.



The following snippet of an API JavaScript Object Notation (JSON) file shows an in-context decoy API:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",
 "url": "/shop",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "login_url": "",
 "api_mapping": {
 "internal_url": ""
 },
;
; Note – other configuration parameters removed
;
 "decoy_config":
 {
 "decoy_enabled": true,
 "response_code" : 200, decoy API Configuration
 "response_def" : "OK",
 "response_message" : "OK",
 "decoy_subpaths": [
 "/shop/root",
 "/shop/admin"
 ]
 }
 }
}
```

The API JSON file defines normal API paths consisting of the path `/shop`. The decoy configuration is enabled for `/shop/root` and `/shop/admin` with the following parameters:

- `decoy_enabled` parameter is set to `true`. If set to `false`, no decoy paths are configured.
- `response_code` is set to `200`. When a decoy sub-path is accessed, return a `200` response.
- `response_def` is set to `OK`. When a decoy sub-path is accessed, return `OK` as the response.

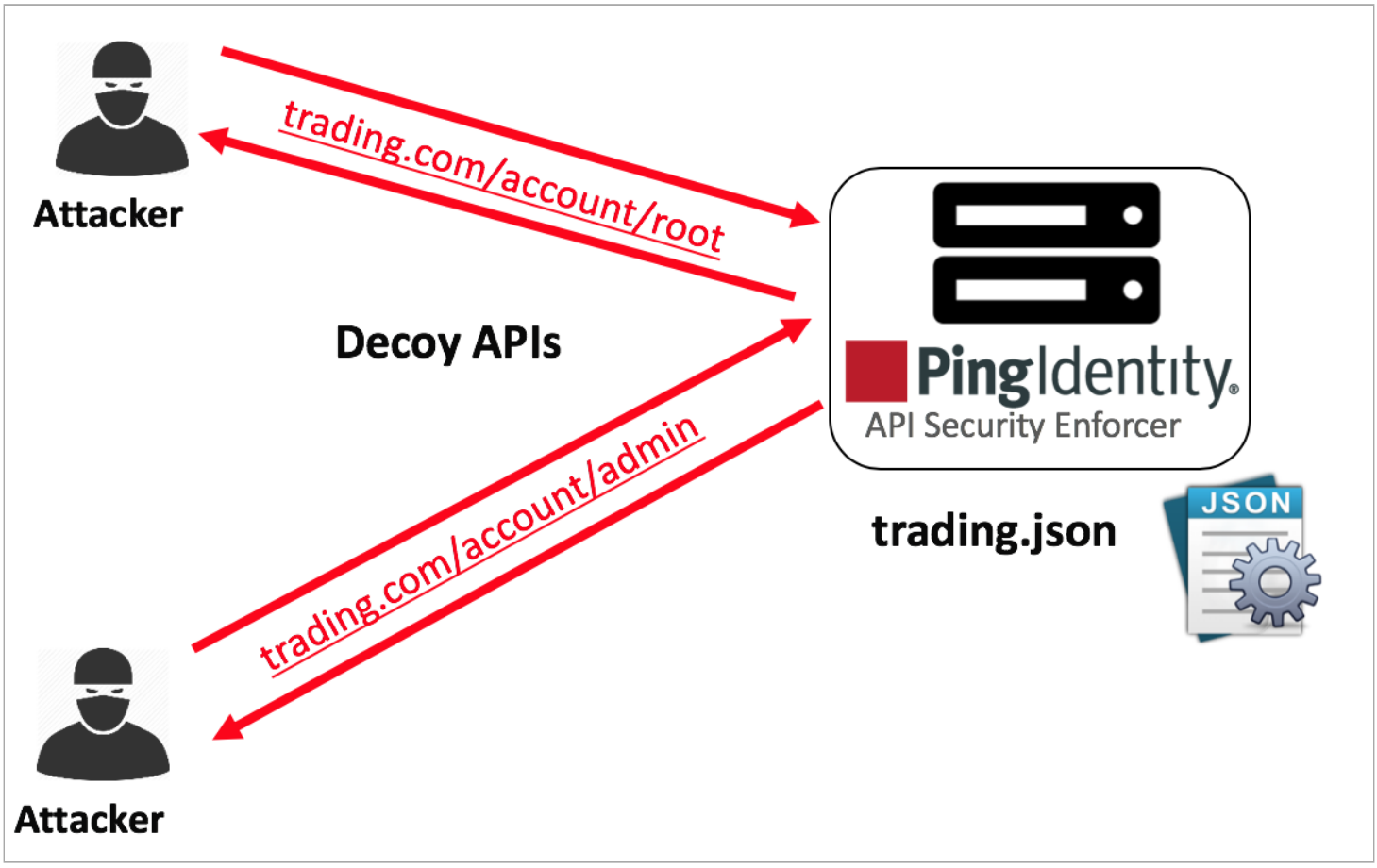
An in-context decoy API can have a maximum of 32 sub-paths configured for an API.

> ⚠ **Warning**
>
> When configuring in-context decoy APIs, do not leave empty sub-paths which makes your business API into an out-of-context API. No traffic will be forwarded to backend application servers.

Out-of-context decoy API

Out-of-context decoy APIs are independent APIs where every path is a decoy API. Any sub-paths accessed in the API are treated as part of the decoy API.

Copyright © 2025 Ping Identity Corporation



The following is a snippet of a trading API JSON file which has been deployed as a decoy API:

```
{
    "api_metadata": {
        "protocol": "http",
        "url": "/account",
        "hostname": "*",
;
; Note – other configuration parameters removed
;
        "decoy_config":
        {
          "decoy_enabled": true,
          "response_code" : 200,
          "response_def" : "OK",
          "response_message" : "OK",          Decoy API Configuration
          "decoy_subpaths": [

          ]
       }
```

Since the `decoy_subpaths` parameter is empty, any sub-path accessed by the attacker after `/account` is regarded as a decoy path or decoy API.

After configuring in-context or out-of-context decoy APIs, you can check the API listings by running the `list_api` command:

Copyright © 2025 Ping Identity Corporation

```
opt/pingidentity/ase/bin/cli.sh list_api -u admin -p
flight ( loaded ), https
shop ( loaded ), https, decoy: in-context
trading ( loaded ), https, decoy: out-context
```

Real-time API deception attack blocking

API Security Enforcer (ASE) detects any client probing a decoy application programming interface (API). When a client probes an out-of-context decoy API, ASE logs but does not drop the client connection. However, if the same client tries to access a legitimate path in the in-context decoy API, then ASE blocks the client in real-time.

Here is a snippet of an ASE access log file showing real-time decoy blocking:

```
[Tue Aug 14 22:51:49:707 2018] [thread:209] [info] [connectionid:1804289383] [connectinfo:
100.100.1.1:36663] [type:connection_drop] [api:decoy] [request_payload_length:0] GET /decoy/test/test HTTP/
1.1
User-Agent: curl/7.35.0
Accept: /
Host: app
```

The blocked client is added to the deny list which can be viewed by running the `view_blacklist` CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist
Realtime Decoy Blacklist
1) type : ip, value : 100.100.1.1
```

ASE DoS and DDoS protection

Application Security Enforcer (ASE) flow control ensures that backend application programming interface (API) servers are protected from unplanned or malicious (for example, DDoS) surges in API traffic. Flow control combines client and backend server traffic control at an API level to protect REST and WebSocket API servers.

Protection for REST APIs

- Client Rate Limiting – Protects against abnormally high traffic volumes from any client (for example, Denial-of-Service - DoS attack). By controlling inbound requests from REST API clients, client rate limiting protects API servers from being overloaded by a single client.
- Aggregate Server TCP Connection Limits – Prevents server overload from too many concurrent TCP connections across one or a cluster of ASE nodes. Restricts the total number of TCP connections allowed from a cluster of ASE nodes to a specific API on each server.
- Aggregate Server HTTP Request Limits – Prevents REST API server overload from too many concurrent HTTP requests across one or a cluster of ASE nodes. Unlike traditional per node flow control, this implementation protects any REST API server from too much aggregate client traffic coming from a cluster of ASE nodes (for example, traffic load bursts or DDoS attacks).
- Client Request Queuing – Queues and retries REST API session requests when servers are busy.

Copyright © 2025 Ping Identity Corporation

Protection for WebSocket APIs

- Client Rate Limiting – Protects against abnormally high traffic volumes from any client (for example, Denial-of-Service - DoS attack). By controlling the client HTTP requests and WebSocket traffic volumes, rate limiting protects API servers from being overloaded by a single client.
- Aggregate Server Connection Limits – Prevents server overload from too many simultaneous session connections across one or a cluster of ASE nodes. Restricts the total number of WebSocket sessions allowed from a cluster of ASE nodes to a specific API on each server.
- Outbound Rate Limiting – Protects against abnormally high traffic volumes to a client. By managing outbound traffic volumes to WebSocket clients, outbound rate limiting protects against exfiltration.

The following table lists the control functions which apply to each protocol.

	REST API (HTTP/HTTPS)	WebSocket and Secure WebSocket
Client Spike Threshold		
Server Connection Quota		
Server Connection Queuing		
Server Spike Threshold		N/A
Bytes-in Threshold	N/A	
Bytes-out Threshold	N/A	

REST API protection from DoS and DDoS

Flow control protects REST application programming interface (API) servers from denial-of service (DoS) and distributed denial-of-service (DDoS) attacks using four control variables, which are independently configured. By default, no flow control is enabled.

Copyright © 2025 Ping Identity Corporation

The following table shows the control variables that are configured once in every API JavaScript Object Notation (JSON) file.

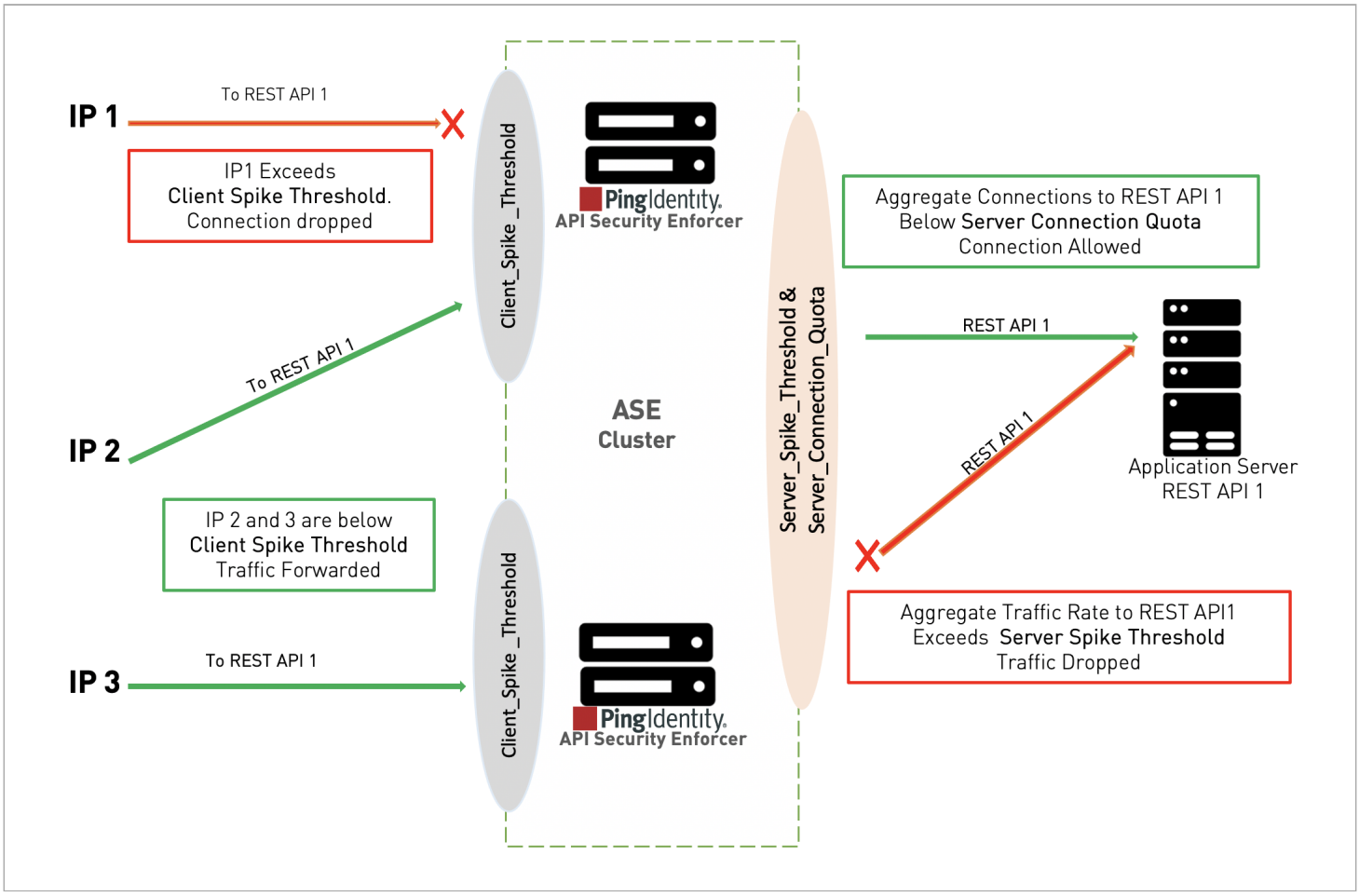
Variable	Description
`client_spike_threshold`	Maximum requests per time-period from a single client IP to a specific REST API. Time can be in seconds, minutes, or hours.
`server_connection_queueing`	When `true`, queue API connection requests when all backend servers reach server connection quota. The default value is `false`.

The following table shows the control variables that are configured for each server in every API JSON file.

Variable	Description
`server_connection_quota`	Maximum number of concurrent connections to a specific REST API on a server. Prevents aggregate connections from one or a cluster of API Security Enforcer (ASE) nodes from overloading a REST API running on a specific server.
`server_spike_threshold`	Maximum requests per time period to the REST API running on the specified server. Prevents the aggregate request rate from one or a cluster of ASE nodes from overloading a REST API running on a specific server. Time can be in seconds, minutes, or hours.

The following diagram shows the effect of the parameters on traffic flow through ASE to backend servers. In the diagram, client-side flow control is managed by the `client_spike_threshold` parameter and server-side flow control is regulated by a combination of the `server_spike_threshold` and `server_connection_quota` parameters.

Copyright © 2025 Ping Identity Corporation



Client flow control monitors incoming traffic from each client connection and drops the session when traffic limits are exceeded. The diagram shows the following client scenarios:

- IP1 sends request volumes that exceed the `client_spike_threshold` value. ASE 1 sends an error message and terminates the session to stop the attack.
- IP2 and IP3 sends request traffic that stays below the `client_spike_threshold` value. Requests are passed to the backend API servers.

Server-side flow control manages traffic volumes and session count for an API on an application server. `server_connection_quota` sets the maximum number of concurrent connections that can be established to each API on a server. `server_spike_threshold` controls the aggregate traffic rate to an API on a server.

The concurrent connections and request rate consist of the aggregate traffic from all ASE nodes forwarding traffic to an API on a server. The diagram shows two server scenarios:

- A new connection request from ASE 1 is allowed because it's within the `server_connection_quota` threshold.
- ASE 2 detects that the combined traffic rate from ASE 1 and ASE 2 will exceed the `server_spike_threshold` for REST API 1. It drops IP 3 traffic and sends an error message to the client.

Example

The following is an example for an application server that explains the scenarios depicted by the previous diagram.

Copyright © 2025 Ping Identity Corporation

Parameter	Configured value
`client_spike_threshold`	50,000 requests per second per IP
`server_spike_threshold`	30,0000 requests per second per server
`server_connection_quota`	20,000 concurrent connections per server
`server_connection_queueing`	`true`

- Client flow control permits a maximum of 50,000 requests per second from an individual IP. If IP 1, 2, or 3 exceeds the 50,000 per second limit, ASE drops the client session. Otherwise, all requests are passed to the backend servers.
- Server flow control allows 30,000 requests per second to REST API 1 on the application server. If the sum of requests per second from the ASE cluster nodes (ASE 1 + ASE 2 request rate) to REST API1 exceeds 30,000 per second, then traffic is dropped from the client causing aggregate traffic to exceed the maximum request rate. Otherwise, ASE 1 and ASE 2 forward all traffic.
- Server flow control allows 20,000 concurrent connections to REST API1 on the application server. If the sum of connections from the ASE cluster nodes (ASE 1 + ASE 2 connection count) to REST API1 exceeds 20,000, then ASE will queue the request for a time because `server_connection_queuing` is enabled. If queuing is not enabled, then the request is dropped.

Summary table for REST API flow control

Parameter	Notes
`client_spike_threshold`	Maximum request rate from a client to an API
`server_spike_threshold`	Maximum aggregate request rate through ASE cluster nodes to an API on a specific server
`server_connection_quota`	Maximum number of concurrent sessions from ASE cluster nodes to an API on a specific server

> **Note**
>
> You can also configure server connection quota and server spike threshold separately for each backend server.

JSON configuration for REST API flow control

API Security Enforcer (ASE) flow control is configured separately for each API using the API JSON file.

The following example shows the flow control-related definitions in an API JSON file:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",

 "flow_control": {
 "client_spike_threshold": "0/second",
 "server_connection_queueing" : false
 },
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_spike_threshold": "100/second",
 "server_connection_quota": 20
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_spike_threshold": "200/second",
 "server_connection_quota": 40
 }
 ]
 }
}
```

The flow control section includes definitions that apply globally across the API definition and include the `client_spike_threshold` and `server_connection_queueing` parameters. Server specific definitions include the `server_spike_threshold` and `server_connection_quota` parameters, which are configured on each individual server. The default is no flow control with all values set to 0.

You can specify different values for each server for `server_connection_quota` and `server_spike_threshold`.

> **Note**
>
> If server connection quota is set to 0 for one server, then it must be 0 for all other servers in the API JSON definition.

Configuring the flow control CLI for REST API

You can use the API Security Enforcer (ASE) command-line interface (CLI) to update flow control parameters.

About this task

> **Note**
>
> API security must be enabled for ASE flow control to work. For more information on enabling API security, see Enable API security.

Steps

- To update the client spike threshold, run the following command:

Copyright © 2025 Ping Identity Corporation

```
update_client_spike_threshold {api_id} {+ve digit/(second|minute|hour)}
```

Example:

```
update_client_spike_threshold shop_api 5000/second
```

- To update the server spike threshold, run the following command:

```
update_server_spike_threshold {api_id} {host:port} {+ve digit/(second|minute|hour)}
```

Example:

```
update_server_spike_threshold shop_api 5000/second
```

- To update the server connection quota, run the following command:

```
update_server_connection_quota {api_id} {host:port}{+ve digit}
```

Example:

```
update_server_connection_quota shop_api 5000
```

WebSocket API protection from DoS and DDoS

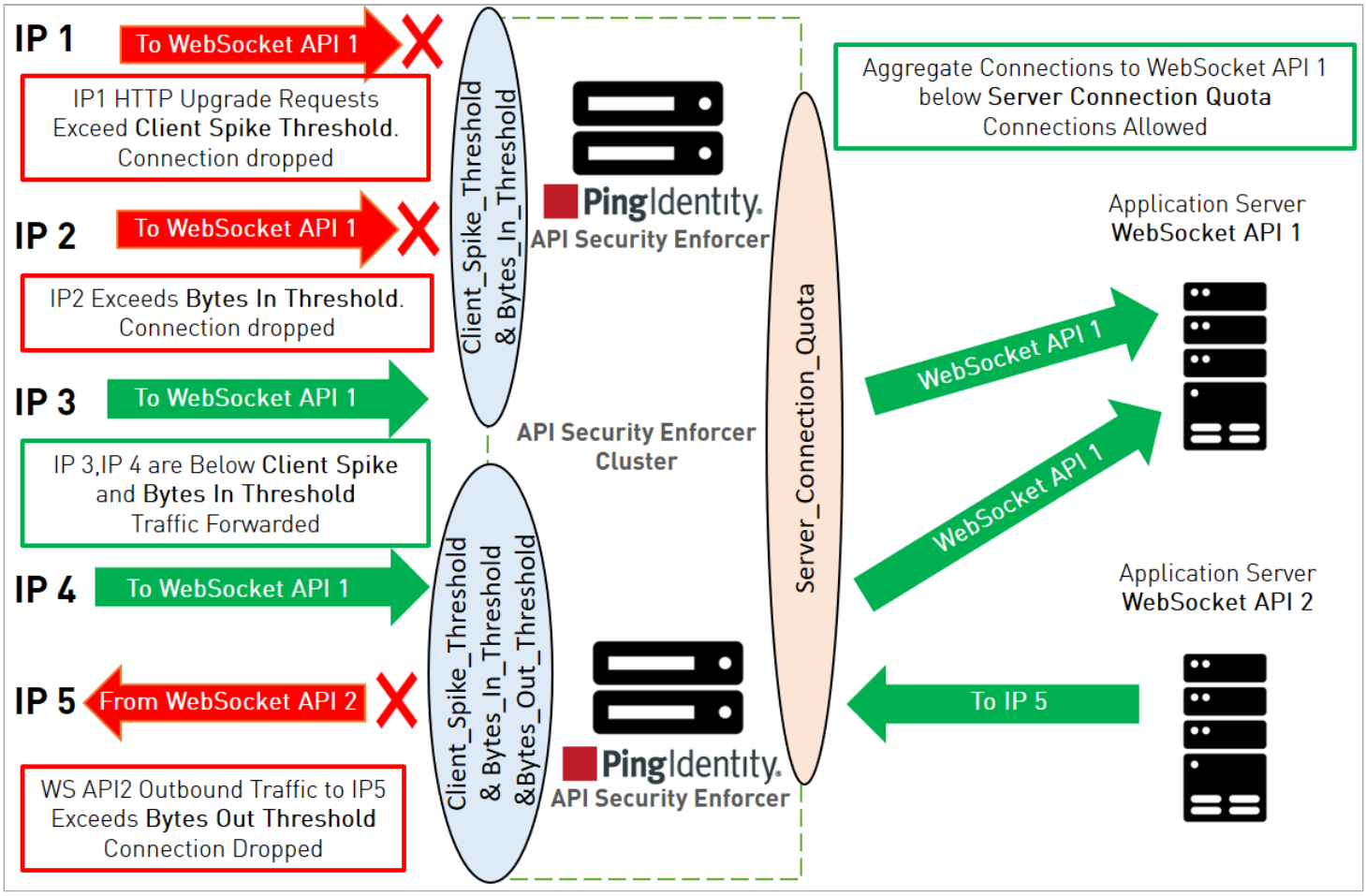
Flow control protects WebSocket (ws) servers using five control variables which are independently configured. By default, no flow control is enabled.

Variable	Description
Configured once in every application programming interface (API) JavaScript Object Notation (JSON) file	
`client_spike_threshold`	Maximum number of HTTP requests per time-period from a single Internet Protocol (IP) to a specific WebSocket API. Time can be in seconds, minutes or hours.
`bytes_in_threshold`	Maximum number of bytes per time-period from a single IP to an API Security Enforcer (ASE) node. Time can be in seconds, minutes or hours.
`bytes_out_threshold`	Maximum number of bytes per time-period sent from an ASE node to a single IP. Time can be in seconds, minutes or hours.
`server_connection_queueing`	When `true`, queue connection requests when all backend servers reach the server connection quota. The default value is `false`.

Copyright © 2025 Ping Identity Corporation

Variable	Description
Configured for each server in every API JSON file	
`server_connection_quota`	**Maximum number of concurrent connections to a specific WebSocket API on a server. Prevents aggregate connections from one or a cluster of ASE nodes from overloading a WebSocket API on a specific server.**

The following diagram shows the effect of the parameters on traffic flow through ASE. In the diagram, client-side flow control is managed by the `client_spike_threshold`, `bytes_in_threshold`, and `bytes_out_threshold` parameters. The `bytes_out` threshold protects against data exfiltration. Server flow control is regulated by the `server_connection_quota` parameter.



Client flow control monitors incoming traffic from each client connection and drops sessions when HTTP request or bytes in threshold limits are exceeded. In addition, outbound traffic from each ASE Node is monitored to protect against exfiltration. The diagram shows client scenarios including:

- **IP1 sending HTTP request volumes which exceed the `client_spike_threshold` value. ASE 1 sends an error message and terminates the session to stop the attack.**
- **IP2 sending WebSocket streaming traffic volumes which exceed the `bytes_in_threshold` limits. ASE 1 sends an error message and terminates the session to stop the traffic.**

Copyright © 2025 Ping Identity Corporation

- IP3 and IP4 within client spike threshold and bytes in threshold criteria and requests are forwarded to the backend server.
- Traffic from ASE 2 to IP5 exceeds the bytes out threshold value. ASE blocks the traffic and drops the client session.

The server-side flow control provides the ability to control session count to an API on an application server. `server_connection_quota` sets the maximum number of concurrent connections that can be established to an API on a server. The concurrent connections are the aggregate connections from all ASE nodes forwarding traffic to the specified API on a given server.

The following table is an example with a hypothetical deployment for the Application Server in the previous diagram.

Variable	Configured value
`client_spike_threshold`	50,000 requests per second per IP
`bytes_in_threshold`	2000 bytes per second per IP
`bytes_out_threshold`	1000 bytes per second per server
`server_connection_quota`	20,000 concurrent connections per server
`[.parmname]` server_connection_queueing````	`true`

Client flow control permits a maximum of 50,000 HTTP requests/second from an individual IP. If IP 1, 2, or 3 exceeds the 50,000/second limit, ASE drops the client session. Otherwise, all requests are passed to the backend servers.

Client flow control allows a maximum of 2,000 bytes/second from each WebSocket client connection to an ASE node. If IP 1, 2, or 3 exceeds the 2,000 bytes/second limit, ASE drops the client session. Otherwise, all requests are passed to the backend servers.

Server flow control allows 20,000 concurrent connections to WebSocket API 1 on the application server. If the sum of connections from the ASE cluster nodes (i.e. ASE 1 + ASE 2 connection count) to WebSocket API1 exceeds 20,000, then ASE will queue the request for a time-period since `server_connection_queuing` is enabled. If queuing is not enabled, then the request is dropped.

Client Flow Control allows a maximum of 1,000 bytes/second from a WebSocket API to any WebSocket client connection. If outbound traffic exceeds the 1,000 bytes/second limit, ASE blocks the traffic and drops the client session. Otherwise, all requests are passed to the backend servers.

The following is a summary table for WebSocket flow control:

Parameter	Notes
`client_spike_threshold`	Maximum HTTP request rate from a client to an API
`bytes_in_threshold`	Maximum number of bytes per time-period from a client to a specific ASE node
`bytes_out_threshold`	Maximum number of bytes per time-period from an ASE node

Copyright © 2025 Ping Identity Corporation

Parameter	Notes
`server_connection_quota`	Maximum number of concurrent sessions from ASE cluster nodes to an API on a specific server

Flow control configuration for WebSocket API

API Security Enforcer (ASE) flow control is configured separately for each API using the API JSON file.

Here are the flow control related definitions in an API JSON file:

```
{
 "api_metadata": {
 "protocol": "ws",

 "flow_control": {
 "client_spike_threshold": "0/second",
 "bytes_in_threshold": "0/second",
 "bytes_out_threshold": "0/second",
 "server_connection_queueing" : false
 },
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_connection_quota": 10
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_connection_quota": 20
 }
 ]
 }
}
```

The flow control section includes definitions which apply globally across all servers running the defined WebSocket API. These are `client_spike_threshold`, `bytes_in_threshold`, `bytes_out_threshold`, and `server_connection_queueing`. Server specific definitions include `server_connection_quota` which is configured on each individual server. The default is no flow control with all values set to zero. Note that different values can be specified for each server for `server_connection_quota`.

> **Note**
>
> If server connection quota is set to zero for one server, then it must be zero for all other servers in the API JSON definition.

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> API security must be enabled for ASE flow control to work. For more information on enabling API security using the configuration file, see Defining an API using API JSON configuration file in inline mode. For more information on using the command-line interface (CLI), see Enabling and disabling real-time API cybersecurity.

Flow control CLI for WebSocket API

ASE CLI can be used to update flow control parameters:

Update Client Spike Threshold:

Enter the following command to update the client spike threshold:

```
update_client_spike_threshold {api_id} {+ve digit/(second|minute|hour)}
```

For example: `update_client_spike_threshold shop_api 5000/second`

Update Bytes-in

```
update_bytes_in_threshold {api_id} {+ve digit/(second|minute|hour)}
```

For example: `update_bytes_in_threshold shop_api 8096/second`

Update Bytes-out

```
update_bytes_out_threshold {api_id} {+ve digit/(second|minute|hour)}
```

For example: `update_bytes_out_threshold shop_api 8096/second`

Update Server Quota

`update_server_connection_quota {api_id} \{host:port}\{+ve digit}`

For example: `update_server_connection_quota shop_api 5000`

> **Note**
>
> API security must be enabled for ASE flow control to work. For more information on enabling API security, see Enabling and disabling real-time API cybersecurity.

Server connection queuing for REST and WebSocket APIs

API Security Enforcer (ASE) can queue server connection requests when the backend application programming interface (API) servers are busy. When enabled, server connection queuing applies to both REST and WebSocket APIs and is configured in the API JavaScript Object Notation (JSON) file.

Copyright © 2025 Ping Identity Corporation

Connection queuing for stateless connections

Stateless connections are connections without cookies. Before enabling connection queuing, configure connection quota values for the backend API servers. After both connection quota and connection queuing are set, the requests are routed based on the following weightage formula:

$$\frac{Q_i}{\sum_{i=1}^{n} Q_i}$$

Where *Q i* is the server connection quota for servers from *i=1 to i=n*

For example, if two backend servers have connection quota set as 20,000 and 40,000 connections, then the connections are served in a ratio of 20000/ (20000+40000) and 40000/ (20000+40000), that is, in the ratio of 1/3 and 2/3 for the respective servers.

When queuing is enabled and the backend servers are occupied, the connections are queued for a period. The connections are forwarded to the next available backend server during the queuing period based on the weighted ratio of server connection quota.

Connection queueing for stateful connections

Stateful connections are connections with cookies. In this mode, cookies are used to establish sticky connections between the client and the server. Before enabling connection queuing, configure connection quota values for the backend API servers. After both connection quota and connection queuing are set, the requests are routed based on the following formula:

$$\frac{Q_i}{\sum_{i=1}^{n} Q_i}$$

Where *Q i* is the server connection quota for servers from *i=1 to i=n*

For example, if two backend servers have connection quota set as 20,000 and 40,000 connections, then the connections are served in a ratio of 20000/ (20000+40000) and 40000/ (20000+40000), that is, in the ratio of 1/3 and 2/3 for the respective servers. The weighted ratio of connection distribution is reached when the server connection quota is reached for all backend servers. Stateful connection distribution considers cookie stickiness with backend servers.

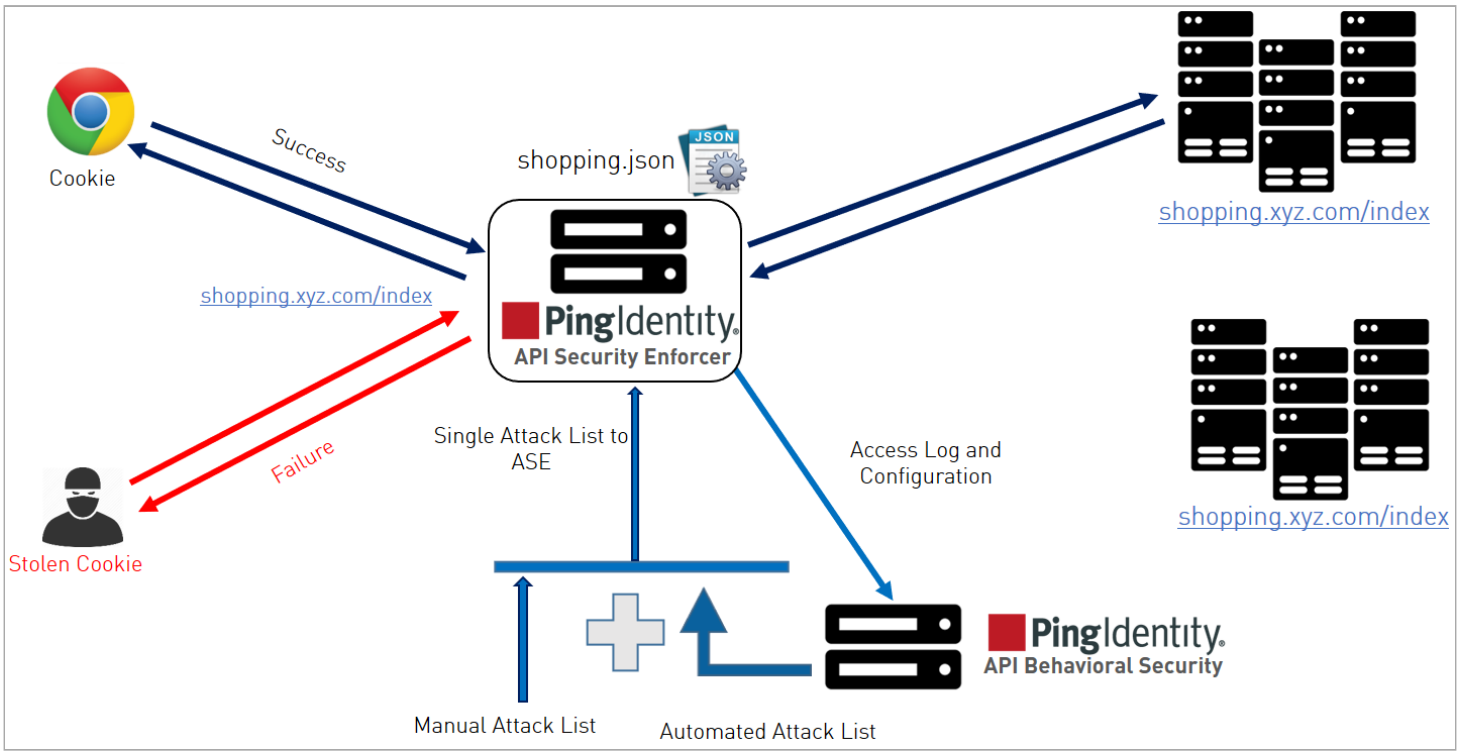
When queuing is enabled and the backend servers are occupied, the connections are queued for a period. Stateful connections are attempted with the same backend server. If the server becomes available during the queuing period, the connections are served. If the backend server is not available, the connections are dropped.

Copyright © 2025 Ping Identity Corporation

ABS AI-based security

The API Behavioral Security (ABS) AI Engine detects attacks using artificial intelligence (AI) algorithms.

After receiving API Security Enforcer (ASE) access logs and application programming interface (API) JavaScript Object Notation (JSON) configuration files, ABS applies AI algorithms to track API connections and detect attacks. If the `enable_abs_attack` parameter is set to `true`, ABS sends the deny list to ASE, which blocks client identifiers, such as API keys, usernames, cookie, Internet Protocol (IP) address, and OAuth token on the list.



ASE to ABS connectivity

To connect the API Security Enforcer (ASE) to API Behavioral Security (ABS), configure the ABS address (IPv4:*<port>* or *<hostname>*:*<port>*), access key, and secret key in the `abs.conf` file located in the `/<ASE installation path>/pingidentity/ase/config` directory.

> **Important**
>
> The `enable_abs` parameter must be set to `true` in the `ase.conf` file. When ABS is in a different AWS security group, use a private IP address.

The following table includes the parameter values and descriptions.

Parameter	Description
`deployment_type`	The ABS deployment mode. Valid values are `cloud` or `onprem`. The default value is `onprem`.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`gateway_credential`	This parameter is used when ABS is deployed in `cloud` mode. The credential generated in PingOne while creating a PingIntelligence connection is assigned here so that PingOne can authenticate the data sent by ASE during runtime. For more information on PingOne connections, see Connections.
`abs_cloud_endpoint`	Use this parameter to assign an endpoint other than the one decoded by the gateway credentials. It's used when ABS is deployed in `cloud` mode.
`abs_endpoint`	The parameter has two possible configurations: • When ABS is deployed with a load balancer - Configure the host name and port or the IPv4 and port of the load balancer. **Note** To allow the load balancer to bind ASE's session to a specific ABS node, enable cookie stickiness in the load balancer with `PISESSIONID` cookie. • When ABS is deployed without a load balancer -Configure the parameter with the host name and port or the IPv4 and port of all the ABS nodes in a cluster. If later a new ABS node is added to the cluster, add its host name or IPv4 to the list and restart ASE.
`access_key`	The access key or the username for the ABS nodes. It is the same for all the ABS nodes. The same value has to be configured in ABS MongoDB database. This value is obfuscated during the start of ASE. This parameter is used when ABS is deployed in `onprem` mode. **Note** ":" is a restricted character and allowed in access key.
`secret_key`	The secret key or the password for the ABS nodes. It is the same for all the ABS nodes. The same value has to be configured in ABS MongoDB database. This value is obfuscated during the start of ASE. This parameter is used when ABS is deployed in `onprem` mode. **Note** ":" is a restricted character and allowed in secret key.
`enable_ssl`	Set the value to true for SSL communication between ASE and ABS. The default value is `true`. ASE sends the access log files in plain text if the value is set to `false`. This parameter is used when ABS is deployed in `onprem` mode.
`abs_ca_cert_path`	Location of the trusted certificate authority (CA) certificates for SSL/TLS connections from ASE to ABS. If the path parameter value is left empty, then ASE does not verify the validity of CA certificates. However, the connection to ABS is still encrypted. This parameter is used when ABS is deployed in `onprem` mode.

The `access_key` and `secret_key` are configured in ABS. For more information, see *ABS Admin Guide*.

Copyright © 2025 Ping Identity Corporation

The following is a sample `abs.conf` file:

```
; API Security Enforcer ABS configuration.;
 This file is in the standard .ini format.
The comments start with a semicolon (;).;
Following configurations are applicable only if ABS is enabled with true.
; Configure ABS deployment type. Supported values (onprem/cloud)
deployment_type=onprem

; PingIntelligence Gateway Credentials
gateway_credential=

; ABS endpoint for cloud
abs_cloud_endpoint=

; a comma-separated list of abs nodes having hostname:port or ipv4:port as an address.
abs_endpoint=127.0.0.1:8080

; access key for abs node
access_key=OBF:AES://ENOzsqOEhDBWLDY+pIoQ:jN6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0

; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw:zzi77GIFF5xkQJccjIrIVWU+RY5CxUhp3NLcNBel+3Q

; Setting this value to true will enable encrypted communication with ABS.
enable_ssl=true

; Configure the location of ABS's trusted CA certificates. If empty, ABS's certificate
; will not be verified
abs_ca_cert_path=
```

Managing ASE blocking of ABS-detected attacks

You can configure the API Security Enforcer (ASE) to automatically fetch and block API Behavioral Security (ABS)-detected attacks.

Enable or disable attack list fetching from ABS:

Enabling attack list fetching from ABS

Steps

1. To enable ASE Security, run the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
   ```

2. To enable ASE to send API traffic information to ABS, run the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs
   ```


Copyright © 2025 Ping Identity Corporation

3. To enable ASE to fetch and block ABS detected attacks, run the following command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs_attack
```

Result:

After enabling automated attack blocking, ASE periodically fetches the attack list from ABS and blocks the identified connections.

4. To set the time interval at which ASE fetches the attack list from ABS, configure the `abs_attack_request_minute` parameter in `ase.conf` file

Example:

```
; This value determines how often ASE will query ABS.
abs_attack_request_minutes=10
```

Disabling attack list fetching from ABS

Steps

- To disable ASE from fetching the ABS attack list, run the following command-line interface (CLI) command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs_attack
```

> **Note**
>
> The ABS attack list can be viewed using ABS APIs and used to manually configured an attack list on ASE. For more information on ABS APIs, see ABS Administration.

Result:

After entering the above command, ASE will no longer fetch the attack list from ABS. However, ABS continues generating the attack list and stores it locally.

- To stop an ASE cluster from sending log files to ABS, run the following ASE CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs
```

> **Note**
>
> For more information on types of attacks, see ABS AI Engine.

Result:

After entering this command, ABS will not receive any logs from ASE.

Copyright © 2025 Ping Identity Corporation

CLI for inline ASE

The following table shows the command-line interface (CLI) functions and their syntax for API Security Enforcer (ASE) in inline mode.

Function	Description	Syntax
Start ASE	Starts ASE	`./start.sh`
Stop ASES	Stops ASE	`./stop.sh`
Help	Displays `cli.sh` help	`./cli.sh help`
Version	Displays the version number of ASE	`./cli.sh version`
Status	Displays the running status of ASE	`./cli.sh status`
Update password	Changes ASE admin password	`./cli.sh update_password \{-u admin}`
Change log level	Change `balancer.log` and `controller.log` log level	`./cli.sh log_level -u admin -p` Options: • `warn` • `info` • `error` • `fatal` • `debug`
Get Authentication Method	Displays the current authentication method	`./cli.sh get_auth_method {method} \{-u admin}`
Update Authentication Method	Updates ASE authentication method	`./cli.sh update_auth_method {method} \{-u admin}`
Enable Audit Logging	Enables audit logging	`./cli.sh enable_audit -u admin -p admin`
Disable Audit Logging	Disables audit logging	`./cli.sh disable_audit -u admin -p admin`
Add Syslog Server	Adds a new syslog server	`./cli.sh –u admin -p admin add_syslog_server host:port`
Delete Syslog Server	Deletes the syslog server	`./cli.sh –u admin -p admin delete_syslog_server host:port`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
List Syslog Server	Lists the current syslog server	`./cli.sh –u admin -p admin list_syslog_server`
Add API	Adds a new API from config file in JSON format. File should have `.json` extension	`./cli.sh –u admin -p admin add_api {config_file_path}`
Update API	Updates an API after the API JSON file has been edited and saved.	`./cli.sh –u admin -p admin update_api {api_name}`
List APIs	Lists all APIs configured in ASE	`./cli.sh –u admin -p admin list_api`
API Info	Displays the API JSON file	`./cli.sh –u admin -p admin api_info {api_id}`
API Count	Displays the total number of APIs configured	`./cli.sh –u admin -p admin api_count`
List API Mappings	Lists all the external and internal URL mappings.	`./cli.sh –u admin -p admin list_api_mappings`
Delete API	Deletes an API from ASE. Deleting an API removes the corresponding JSON file and deletes all the cookies associated with that API	`./cli.sh –u admin -p admin delete_api {api_id}`
Add a Server	Adds a backend server to an API. Provide the IP address and port number of the server	`./cli.sh –u admin -p admin add_server {api_id}\{host:port} [quota][spike_threshold]`
List Server	Lists all servers for an API	`./cli.sh –u admin -p admin list_server {api_id}`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
Delete a Server	Deletes a backend server from an API. Provide the IP address and port number of the serve	`./cli.sh –u admin -p admin delete_server {api_id}\{host:port}`
Enable Per API Blocking	Enables attack blocking for the API	`./cli.sh –u admin -p admin enable_blocking {api_id}`
Disable Per API Blocking	Disables attack blocking for the API	`./cli.sh –u admin -p admin disable_blocking {api_id}`
Enable Health Check	Enables health check for a specific API	`./cli.sh -u admin -p admin enable_health_check shop_api`
Disable Health Check	Disables health check for a specific API	`./cli.sh -u admin -p admin disable_health_check {api_id}`
Generate Master Key	Generates the master obfuscation key `ase_master.key`	`./cli.sh -u admin -p admin generate_obfkey`
Obfuscate Keys and Password	Obfuscates the keys and passwords configured in various configuration files	`./cli.sh -u admin -p admin obfuscate_keys`
Create a Key Pair	Creates private key and public key pair in key store	`./cli.sh –u admin -p admin create_key_pair`
Create a CSR	Creates a certificate signing request	`./cli.sh –u admin -p admin create_csr`
Create a Self-Signed Certificate	Creates a self-signed certificate	`./cli.sh –u admin -p admin create_self_sign_cert`
Import Certificate	Imports CA-signed certificate into key store	`./cli.sh –u admin -p admin import_cert {cert_path}`
Create Management Key Pair	Creates a private key for management server	`/cli.sh –u admin -p admin create_management_key_pair`
Create Management CSR	Creates a certificate signing request for management server	`/cli.sh –u admin -p admin create_management_csr`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
Create Management Self-signed Certificate	Creates a self-signed certificate for management server	`/cli.sh –u admin -p admin create_management_self_sign_cert`
Import Management Key Pair	Imports a key-pair for management server	`/cli.sh –u admin -p admin import_management_key_pair {key_path}`
Import Management Certificate	Imports CA-signed certificate for management server	`/cli.sh –u admin -p admin import_management_cert {cert_path}`
Health Status	Displays health status of all backend servers for the specified API	`./cli.sh –u admin -p admin health_status {api_id}`
Cluster Info	Displays information about an ASE cluster	`./cli.sh –u admin -p admin cluster_info`
Server Count	Lists the total number of APIs associated with an API	`./cli.sh –u admin -p admin server_count {api_id}`
Cookie Count	Lists the live cookie count associated with an API	`./cli.sh –u admin -p admin cookie_count {api_id}`
Persistent Connection Count	Lists the WebSocket or http-keep alive connection count for an API	`./cli.sh –u admin -p admin persistent_connection_count {api_id}`
Clear cookies	Clears all cookies for an API	`./cli.sh –u admin -p admin clear_cookies{api_id}`
Enable Firewall	Enables API firewall. Activates pattern enforcement, API name mapping, manual attack type	`./cli.sh –u admin -p admin enable_firewall`
Disable Firewall	Disables API firewall	`./cli.sh –u admin -p admin disable_firewall`
Enable ASE detected attacks	Enables ASE detected attacks	`./cli.sh –u admin -p admin enable_ase_detected_attack`
Disable ASE Detected Attacks	Disables API firewall	`./cli.sh –u admin -p admin disable_ase_detected_attack`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax
Enable ABS	Enables ABS to send access logs to ABS	`./cli.sh –u admin -p admin enable_abs`
Disable ABS	Disables ABS to stop sending access logs to ABS	`./cli.sh –u admin -p admin disable_abs`
Enable ABS Detected Attack Blocking	Enables ASE to fetch ABS detected attack lists and block access of list entries.	`./cli.sh –u admin -p admin enable_abs_attack`
Disable ABS Detected Attack Blocking	Stops ASE from blocking and fetching ABS detected attack list. This command does not stop ABS from detecting attacks.	`./cli.sh –u admin -p admin disable_abs_attack`

Adding deny list	Adds an entry to ASE deny list using CLI. Valid type values are: IP, Cookie, OAuth2 token, API Key, and username	`./cli.sh –u admin -p admin add_blacklist {type}{name}{value}`
		If type is `ip`, then `name` is the IP address.
		If type is `cookie`, then `name` is the cookie name, and `value` is the cookie value
		Example:
		`/cli.sh -u admin -p admin add_blacklist ip 1.1.1.1`

Delete deny list Entry	Deletes entry from the deny list.	`./cli.sh –u admin -p admin delete_blacklist {type}{name}{value}`
		Example:
		`cli.sh -u admin -p delete_blacklist token 58fcb0cb97c54afbb88c07a4f2d73c35`

Clear deny list	Clears all the entries from the deny list	`./cli.sh –u admin -p admin clear_blacklist`
View deny list	Views the entire deny list or view a deny list for the specified attack type (for example, invalid_method)	`./cli.sh –u admin -p admin view_blacklist \{all|manual|abs_generated|invalid_content_type|invalid_method|invalid_protocol|decoy|missing_token}`
View deny list for IP addresses with missing tokens	Views the deny list entries that are blocked due to missing tokens	`./cli.sh view_blacklist missing_token -uadmin -padmin`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax

Adding allow list	Adds an entry to ASE allow list using CLI. Valid type values are: IP, cookie, OAuth2 token, API key, and username	`./cli.sh –u admin -p admin add_whitelist {type}{name}{value}`
		Options:
		• If type is `ip`, then `name` is the IP address.
		• If type is `cookie`, then `name` is the cookie name, and `value` is the cookie value
		Example:
		`/cli.sh -u admin -p admin add_whitelist api_key AccessKey 065f73cdf39e486f9d7cda97d2dd1597`

Delete allow list entry	Deletes entry from the allow list	`./cli.sh –u admin -p admin delete_whitelist {type}{name}{value}`
		Example:
		`/cli.sh -u admin -p delete_whitelist token 58fcb0cb97c54afbb88c07a4f2d73c35`

Clear allow list	Clears all the entries from the allow list	`./cli.sh –u admin -p admin clear_whitelist`
View allow list	Views the entire allow list	`./cli.sh –u admin -p admin view_whitelist`
ABS Info	Displays ABS status information. ABS enabled or disabled, ASE fetching ABS attack types, and ABS cluster information	`./cli.sh –u admin -p admin abs_info`
Enable XFF	Enables X-Forwarded For	`./cli.sh –u admin -p admin enable_xff`
Disable XFF	Disables X-Forwarded For	`./cli.sh –u admin -p admin disable_xff`

Update Client Spike	Update Client Spike Threshold	`update_client_spike_threshold {api_id} \{+ve digit/(second|minute|hour)}`
		Example:
		`update_client_spike_threshold shop_api 5000/second`

Update Server Spike	Updates Server Spike Threshold
	`"*"` - use the same value for all servers
		`update_server_spike_threshold {api_id} \{host:port} \{+ve digit/(second|minute|hour)}`
		Examples:
		`update_server_spike_threshold shop_api 127.0.0.1:9090 5000/second`
		`update_server_spike_threshold shop_api "*" 5000/second`

Copyright © 2025 Ping Identity Corporation

Function	Description	Syntax

Update Bytes-in	Updates bytes in value for a WebSocket API	`update_bytes_in_threshold {api_id} \{+ve digit/(second|minute|hour)}`
		Example:
		`update_bytes_in_threshold shop_api 8096/second`

Update Bytes-out	Updates bytes out value for a WebSocket API	`update_bytes_out_threshold {api_id} \{+ve digit/(second|minute|hour)}`
		Example:
		`update_bytes_out_threshold shop_api 8096/second`

Update Server Quota	Updates the number of API connections allowed on a backend server
	`"*"` - use the same value for all backend servers
		`update_server_connection_quota {api_id} \{host:port} \{+ve digit}`
		Examples:
		`update_server_connection_quota shop_api 127.0.0.1:9090 5000`
		`update_server_connection_quota shop_api "*" 5000`

ASE REST APIs using Postman

For the Postman application, Ping Identity provides two set of Postman collections which are used by Postman to access the API Security Enforcer (ASE) REST application programming interface (API) JavaScript Object Notation (JSON) information. The collections are for ASE in inline or sideband mode.

Multiple options are available for accessing the ASE REST API reporting including:

- Postman App
- Java, Python, C Sharp, or similar languages.
- Java client program (such as Jersey)
- C Sharp client program (such as RestSharp)

> **Important**
>
> Make sure to install Postman 6.2.5 or later.

Disabling ASE self-signed certificates with Postman

API Security Enforcer (ASE) ships with a self-signed certificate. If you want to use Postman with the self-signed certificate of ASE, disable the certificate verification option from Postman's settings.

About this task

To disable Postman from certificate verification:

Copyright © 2025 Ping Identity Corporation

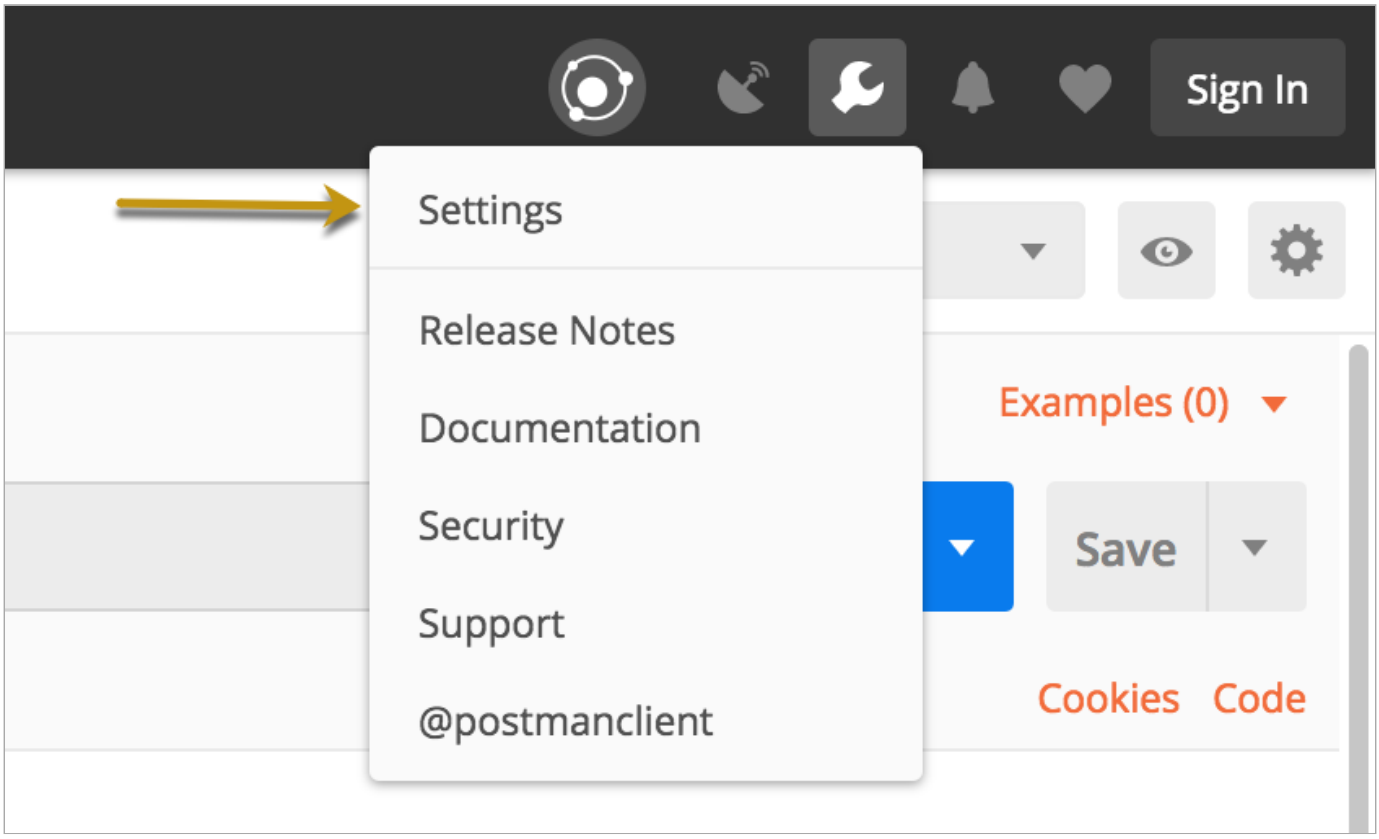
Steps

1. Click the Wrench icon in the top-right corner of the Postman client.

   Result:

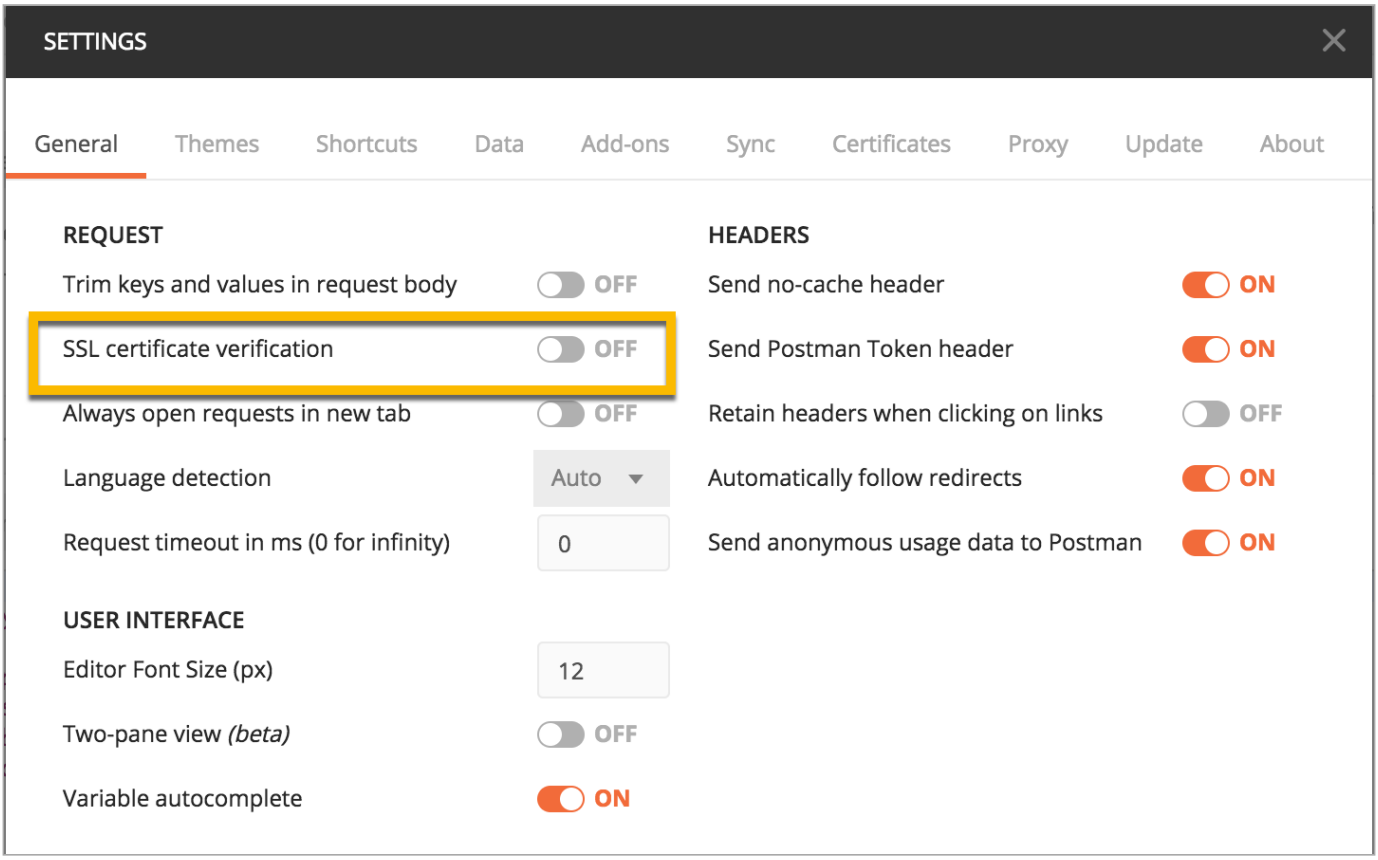
   You see a drop-down list.

2. Select Settings from the list.



3. In the Settings window, to disable SSL certificate verification, in the SSL certificate verification row of the Request list, click the toggle.

Copyright © 2025 Ping Identity Corporation



When the setting is disabled, you see the word Off.

Viewing ASE REST APIs in Postman

View API Security Enforcer (ASE) REST application programming interface (API) reports in Postman.

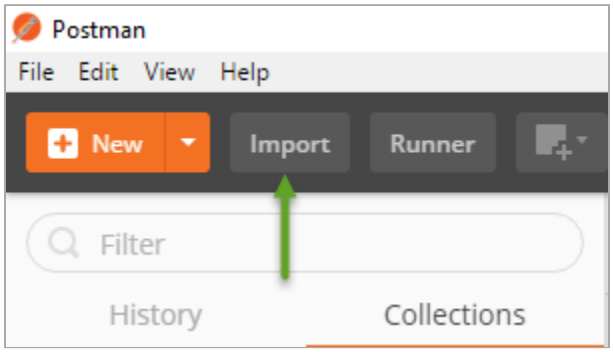
Steps

1. Download the `ASE_4.3_Inline` or `ASE_4.3_Sideband` and `ASE_4.3_Environment` JSON files from the Ping Identity Download site.

   > **Note**
   >
   > These configuration files are used by Postman.

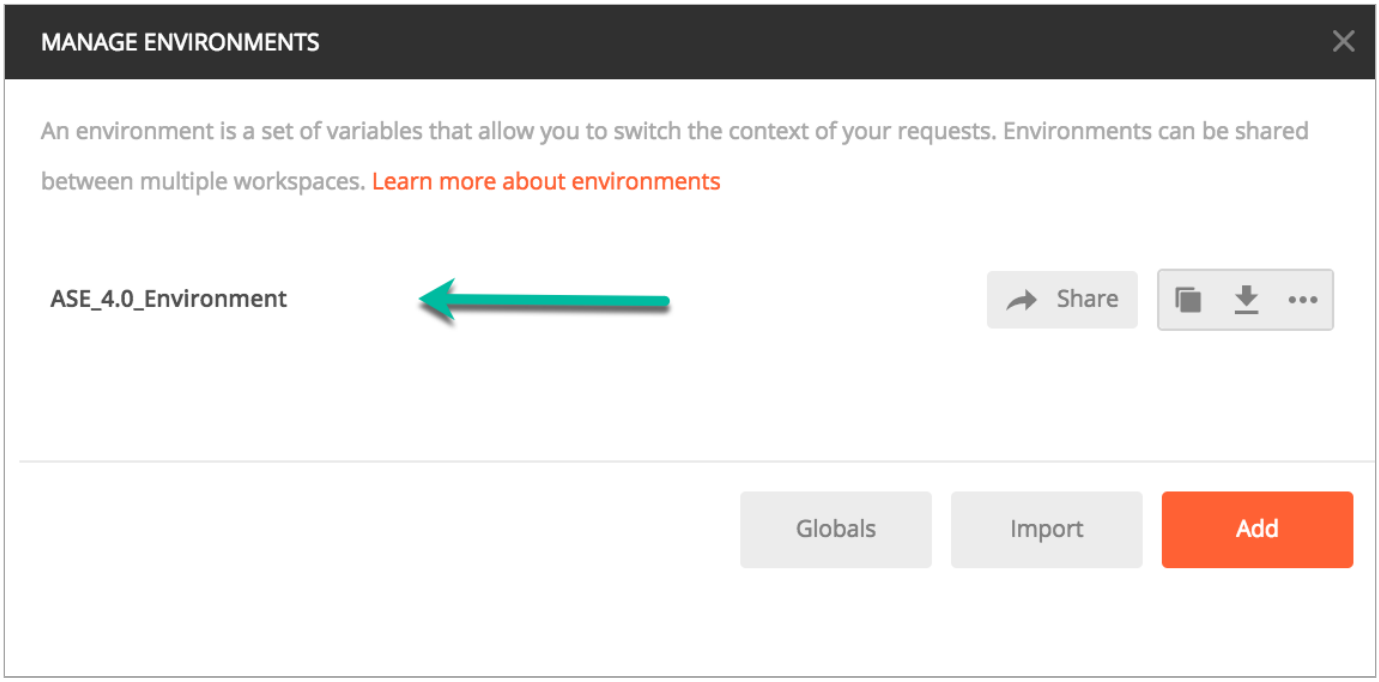
2. Download and install the Postman application 6.2.5 or later.
3. In Postman, to import the two Ping Identity files that you downloaded in step 1, click the Import button.

Copyright © 2025 Ping Identity Corporation



4. After importing the files, click the Gear icon in the upper right corner.

5. In the Manage Environments window, click ASE_4.3_Environment.



6. In theManage Environmentswindow, configure the following values, and then click Update.

Variable	Description
ASE_IP	IP address of the ASE node.
Port	Port number of the ASE node.
Access_Key_Header	Use the default values.
Secret_Key_Header	Use the default values.
Access_Key	Use admin for access key and secret key. If you have changed the admin password, use the updated one.

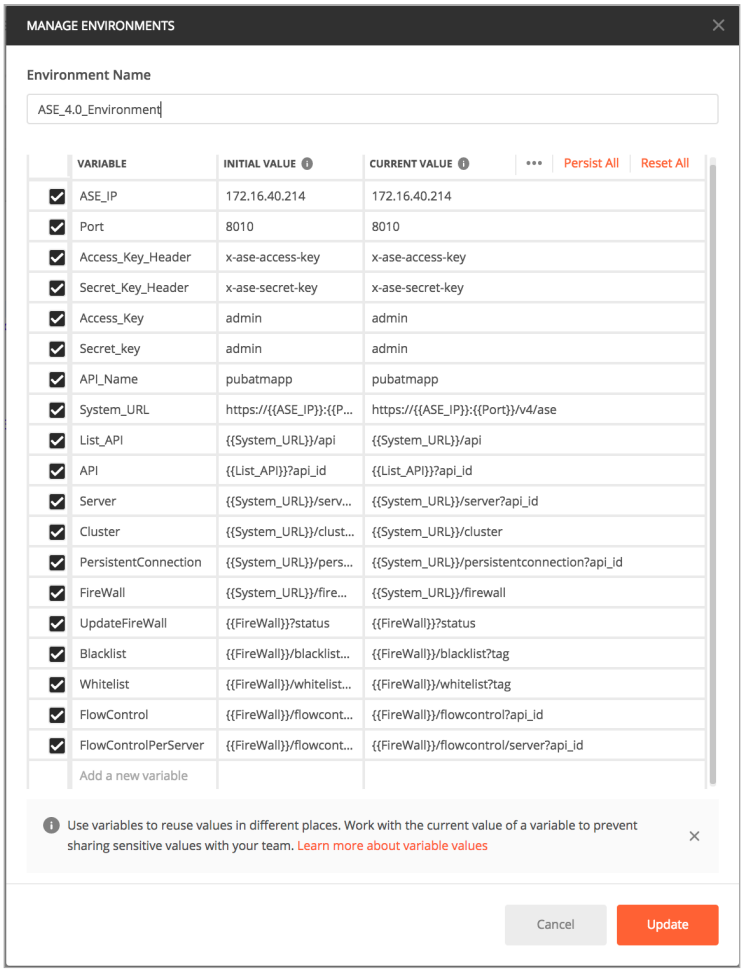
Copyright © 2025 Ping Identity Corporation

Variable	Description
Secret_Key	Use admin for access key and secret key. If you have changed the admin password, use the updated one.
API_Name	The name of the API which you want to administer.

> **Note**
>
> Do not edit any fields that start with the word `System`.

Example:

The following image shows what the Manage Environments window looks like after you set new values, shown in the Current Value column.



7. **In the main Postman window, select the report to display on the left column. Click Send.**

Copyright © 2025 Ping Identity Corporation

REST API for inline and sideband ASE

API Security Enforcer (ASE) REST application programming interface (API) allows you to add, remove, and modify your backend servers.

The REST API payload uses a JavaScript Object Notation (JSON) format. REST API also helps integrate ASE with third-party products. The default port for ASE REST API is `8010`.

The following is a list of formats for ASE's REST APIs:

- Create API (POST) – Inline and sideband ASE
- Read API (GET) – Inline and sideband ASE
- List API (GET) – Inline and sideband ASE
- Update API (PUT) – Inline and sideband ASE
- Create server (POST) – Inline ASE
- Read server (GET) – Inline ASE
- Delete server (DELETE) – Inline ASE
- Read cluster (GET) – Inline ASE
- Read persistent connections (GET) – Inline ASE
- Read firewall status (GET) – Inline and sideband ASE
- Update firewall status (POST) – Inline and sideband ASE
- Add attack type to deny list (POST) – Inline and sideband ASE
- Delete attack type from the allow list (DELETE) – Inline and sideband ASE
- Clear the deny list (DELETE) – Inline and sideband ASE
- View deny list (GET) – Inline and sideband ASE
- Add attack type to allow list (POST) – Inline and sideband ASE
- Delete attack type from the allow list (DELETE) – Inline and sideband ASE
- Clear allow list (DELETE) – Inline and sideband ASE
- View allow list (POST) – Inline and sideband ASE
- Read flow control of an API (GET)– Inline ASE
- Update flow control for an API (POST) – Inline ASE
- Update flow control for a server of an API (POST) – Inline ASE

Copyright © 2025 Ping Identity Corporation

Common request headers

Header	Value
`x-ase-access-key`	`admin` **Note** The default and only allowed access key is `admin`.
`x-ase-secret-key`	`<Secret Key>` **Note** The default secret key is `admin`. You can change the default secret key using the **`update_passowrd`** command.
`Accept`	`application/json`

Create API (POST)

Request

`POST`	`/v4/ase/api?api_id=sample_api`
`Content-Type`	`application/json`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

REST API request:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",
 "url": "/your_rest_api",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "401",
 "error_def": "Unauthorized",
 "error_message_body": "401 Unauthorized"
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "128mb",
 "health_check": true,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/health",
 "server_ssl": false,
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_spike_threshold": "0/second",
 "server_connection_quota": 0
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_spike_threshold": "0/second",
 "server_connection_quota": 0
 }
 ],
 "decoy_config": {
 "decoy_enabled": false,
 "response_code": 200,
 "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
  "response_message": "",
  "decoy_subpaths": []
  }
  }
}
```

WebSocket API request:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "ws",
 "url": "/your_websocket_api",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "401",
 "error_def": "Unauthorized",
 "error_message_body": "401 Unauthorized"
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "bytes_in_threshold": "0/second",
 "bytes_out_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "128mb",
 "health_check": true,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/health",
 "server_ssl": false,
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_connection_quota": 0
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_connection_quota": 0
 }
 ],
 "decoy_config": {
 "decoy_enabled": false,
 "response_code": 200,
 "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
  "response_message": "",
  "decoy_subpaths": []
  }
  }
 }
```

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`{“status” : “success” , “status_message” : “success” }`
`403`	`fail`	`{“status” :”api_already_exists” ,”status_message” :”api sample_api already exists”}`
`403`	`fail`	`{“status” : “validation_error” , “status_message” : “<detailed validation error description” }`

Read API (GET)

Request

`GET`	`/v4/ase/api?api_id=sample_api`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

Response

Copyright © 2025 Ping Identity Corporation

`200`	`success`	**REST API:**

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",
 "url": "/your_rest_api",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "401",
 "error_def": "Unauthorized",
 "error_message_body": "401 Unauthorized"
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "128mb",
 "health_check": true,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/health",
 "server_ssl": false,
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_spike_threshold": "0/second",
 "server_connection_quota": 0
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_spike_threshold": "0/second",
 "server_connection_quota": 0
 }
 ],
 "decoy_config": {
 "decoy_enabled": false,
 "response_code": 200,
 "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
  "response_message": "",
  "decoy_subpaths": []
  }
  }
 }
```

WebSocket API:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "ws",
 "url": "/your_websocket_api",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "401",
 "error_def": "Unauthorized",
 "error_message_body": "401 Unauthorized"
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "bytes_in_threshold": "0/second",
 "bytes_out_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "128mb",
 "health_check": true,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/health",
 "server_ssl": false,
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_connection_quota": 0
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_connection_quota": 0
 }
 ],
 "decoy_config": {
 "decoy_enabled": false,
 "response_code": 200,
 "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

HTTP Code	Status	Content body (application/JSON)

		<pre>"response_message": "",
		"decoy_subpaths": []
		}
		}
		}</pre>

404	not found	<pre>{“status” :”api_not_found” ,”status_message” :”api sample_api does
		not exist”}</pre>

List API (GET)

Request

GET	/v4/ase/api
x-ase-access-key	<Access Key>
x-ase-secret-key	<Secret Key>
Accept	application/json

Response

HTTP Code	Status	Content body (application/JSON)

200	success	<pre>{
		"api_count": "1",
		"api": [
		{
		"api_id": "sample_api",
		"status": "loaded"
		}
		]
		}</pre>

404	not found	<pre>{“status” :”api_not_found” ,”status_message” :”api sample_api does
		not exist”}</pre>

Copyright © 2025 Ping Identity Corporation

Update API (PUT)

Request

`PUT`	`/v4/ase/api?api_id=sample_api`
`Content-Type`	`application/json`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

REST API request:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "http",
 "url": "/your_rest_api",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "401",
 "error_def": "Unauthorized",
 "error_message_body": "401 Unauthorized"
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "128mb",
 "health_check": true,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/health",
 "server_ssl": false,
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_spike_threshold": "0/second",
 "server_connection_quota": 0
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_spike_threshold": "0/second",
 "server_connection_quota": 0
 }
 ],
 "decoy_config": {
 "decoy_enabled": false,
 "response_code": 200,
 "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
    "response_message": "",
    "decoy_subpaths": []
    }
    }
  }
```

WebSocket API request:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_metadata": {
 "protocol": "ws",
 "url": "/your_websocket_api",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "401",
 "error_def": "Unauthorized",
 "error_message_body": "401 Unauthorized"
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "bytes_in_threshold": "0/second",
 "bytes_out_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "128mb",
 "health_check": true,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/health",
 "server_ssl": false,
 "servers": [
 {
 "host": "127.0.0.1",
 "port": 8080,
 "server_connection_quota": 0
 },
 {
 "host": "127.0.0.1",
 "port": 8081,
 "server_connection_quota": 0
 }
 ],
 "decoy_config": {
 "decoy_enabled": false,
 "response_code": 200,
 "response_def": "",
```


Copyright © 2025 Ping Identity Corporation

```
  "response_message": "",
  "decoy_subpaths": []
  }
  }
 }
```

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`` `[.codeph] success` ``	`{“status” : “success” , “status_message” : “success” }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample_api does not exist”}`

Delete API (DELETE)

Request

`DELETE`	`/v4/ase/api?api_id=sample_api`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`{“status” : “success” , “status_message” : “success” }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample_api does not exist”}`

Copyright © 2025 Ping Identity Corporation

Create server (POST)

Request

`POST`	`/v4/ase/server?api_id=<api>`
`Content-Type`	`application/json`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

REST API request:

```
{
 "server":
 {
 "host": "192.168.1.100",
 "port": 8080,
 "server_spike_threshold": "1/second",
 "server_connection_quota": 100
 }
}
WebSocket API Request
{
 "server":
 {
 "host": "192.168.1.100",
 "port": 8080,
 "server_connection_quota": 100
 }
}
```

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`{“status” : “success” , “status_message” : “success” }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample_api does not exist”}`

Copyright © 2025 Ping Identity Corporation

HTTP Code	Status	Content body (application/JSON)
`403`	`fail`	`{“status” : “validation_error” , “status_message” : “detailed info about validation error”}`
`403`	`fail`	`{“status” : “server_exists” , “status_message” :”server already exists”}`

Read server (GET)

Request

`GET`	`/v4/ase/server?api_id=<api_id>`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	**REST API:** `{ “api_id” : “sample_api” “server_count” : 2, “server”: [{ “host” : “192.168.1.100” “port” : 8080, "server_connection_quota": 1000, "server_spike_threshold": "10/second", “health_status” :”Up” }, { “host” : “192.168.1.100” “port” : 8081, server_connection_quota": 1000, "server_spike_threshold": "10/second", “health_status” :”Down” }] }` **WebSocket API:** `{ “api_id” : “sample_api” “server_count” : 2, “server”: [{ "host" : “192.168.1.100” “port” : 8080, "server_connection_quota": 1000, “health_status” :”Up” }, { "host" : “192.168.1.100” “port” : 8081, "server_connection_quota": 1000, “health_status” :”Down” }] }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample_api does not exist”}`

Delete server (DELETE)

Request

`DELETE`	`/v4/ase/server?api_id=<api>`

Copyright © 2025 Ping Identity Corporation

`Content-Type`	`application/json`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

```
{
 “server”:
 {
 "host" : “192.168.1.100”,
 "port" : 8080
 }
}
```

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`{“status” : “success” , “status_message” : “success” }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample_api does not exist”}`
`404`	`fail`	`{“status” :”server_not_found” ,”status_message” :”server does not exist”}`
`403`	`fail`	`{“status” : “validation_error” , “status_message” : “detailed info about json validation error”}`

Read cluster (GET)

Request

`GET`	`/v4/ase/cluster`

Copyright © 2025 Ping Identity Corporation

`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Response

HTTP Code	Status	Content body (application/JSON)

`200`	`success`	{
		“cluster_id” : “test_cluster”
		“node_count” : 2
		, “node”:
		[
		{
		"host" : “192.168.2.100”
		“port” : 8080
		“uuid” : “1c359368-22b6-4713-a5be-15e5cbbddf7a”
		“status” :”active”
		},
		{
		"host" : “192.168.2.101”
		“port” : 8080
		“uuid” : “2d359368-20b6-4713-a5be-15e5cbbde8d”
		“status” :”inactive”
		}
		]
		}

`404`	`fail`	{“status” :”no_cluster_mode” ,”status_message” :”ase is not in cluster mode”}

Read persistent connections (GET)

Request

`GET`	`/v4/ase/persistentconnection?api_id=sample`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`{ “api_id” : “sample” “persistent_connection_count” : { “ws”:1, “wss”:0 } }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample does not exist”}`

Read firewall status (GET)

Request

`GET`	`/v4/ase/firewall`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`\{ “status” :”enabled/disabled”, ”status_message” :”Ok” }`

Update firewall status (POST)

Request

`POST`	`/v4/ase/firewall?status=enable/disable`
`x-ase-access-key`	`<Access Key>`

Copyright © 2025 Ping Identity Corporation

`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Response

HTTP Code	Status	Content body (application/JSON)

`200`	`success`	**If there is a status change:**
		`{ “status” :”enabled/disabled”, ”status_message” :”Firewall is now enabled/disabled” }`
		If there is no change in status:
		`{ “status” :”enabled/disabled”, ”status_message” :”Firewall is already enabled/disabled” }`

`403`	`fail`	`{“status” :”invalid_value” ,”status_message” :”query parameter status contains invalid value”}`

Add attack type to deny list (POST)

Request

`POST`	`/v4/ase/firewall/blacklist`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

```
===============for IP===============
{
 "type" : "ip",
 "value" : "1.1.1.1"
}
===============for Token=============
{
 "type" : "token",
 "value" : "sadjhasiufgkjdsbfkgfa"
}
=============for Cookie/api_key=======
{
 "type" : "cookie/token/api_key",
 "name" : "JSESSIONID",
 "value" : "ljkhasioutfdqbjsfdmakhflia"
}
```

Response

Status code	Response body
`200 OK`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **added to blacklist**
`403 Forbidden`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **already exist**
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`403 Forbidden`	`json parsing error`
`500 Internal Server Error`	`unknown error`

Delete attack type to deny list (DELETE)

Request

`DELETE`	`/v4/ase/firewall/blacklist`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`

Copyright © 2025 Ping Identity Corporation

`Accept`	`application/json`

```
===============for IP===============
{
 "type" : "ip",
 "value" : "1.1.1.1"
}
===============for Token=============
{
 "type" : "token",
 "value" : "sadjhasiufgkjdsbfkgfa"
}
=============for Cookie/api_key=======
{
 "type" : "cookie/token/api_key",
 "name" : "JSESSIONID",
 "value" : "ljkhasioutfdqbjsfdmakhflia"
}
```

Response

Status code	Response body
`200 OK`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **deleted from blacklist**
`403 Forbidden`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **already exist**
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`403 Forbidden`	`json parsing error`
`500 Internal Server Error`	`unknown error`

Clear the deny list (DELETE)

Request

`DELETE`	`/v4/ase/firewall/blacklist?tag=all`
`x-ase-access-key`	`<Access Key>`

Copyright © 2025 Ping Identity Corporation

`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Response

Status code	Response body
`200 OK`	`Blacklist cleared`
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`500 Internal Server Error`	`unknown error`

View deny list (GET)

Request

`GET`	`/v4/ase/firewall/blacklist?tag=`
`Tags`	**Note** Default is `tag=all`. • `all` • `manual` • `abs_generated` • `invalid_content_type` • `invalid_method` • `invalid_protocol` • `decoy`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

Response

Status code	Response body

`200 OK`	{
	"manual_blacklist" : [
	{
	"type" : "cookie",
	"name" : "JSESSIONID",
	"value" : "ljkhasiosalia",
	},
	{
	"type" : "ip",
	"value" : "1.1.1.1",
	}
	],
	"abs_generated_blacklist" : [
	{
	"type" : "cookie",
	"name" : "JSESSIONID",
	"value" : "ljkhasisadosalia",
	},
	{
	"type" : "ip",
	"value" : "1.1.1.2",
	}
	]
	}

`403 Forbidden`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **already exist**
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`500 Internal Server Error`	`unknown error`

Add attack type to allow list (POST)

Request

`POST`	`/v4/ase/firewall/whitelist`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`

Copyright © 2025 Ping Identity Corporation

`Accept`	`application/json`

```
===============for IP===============
{
 "type" : "ip",
 "value" : "1.1.1.1"
}
===============for Token=============
{
 "type" : "token",
 "value" : "sadjhasiufgkjdsbfkgfa"
}
=============for Cookie/api_key=======
{
 "type" : "cookie/token/api_key",
 "name" : "JSESSIONID",
 "value" : "ljkhasioutfdqbjsfdmakhflia"
}
```

Response

Status code	Response body
`200 OK`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **added to whitelist**
`403 Forbidden`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **already exist**
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`403 Forbidden`	`json parsing error`
`500 Internal Server Error`	`unknown error`

Delete attack type from the allow list (DELETE)

Request

`DELETE`	`/v4/ase/firewall/whitelist`
`x-ase-access-key`	`<Access Key>`

Copyright © 2025 Ping Identity Corporation

`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

```
===============for IP===============
{
 "type" : "ip",
 "value" : "1.1.1.1"
}
===============for Token=============
{
 "type" : "token",
 "value" : "sadjhasiufgkjdsbfkgfa"
}
=============for Cookie/api_key=======
{
 "type" : "cookie/token/api_key",
 "name" : "JSESSIONID",
 "value" : "ljkhasioutfdqbjsfdmakhflia"
}
```

Response

Status code	Response body
`200 OK`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **added to whitelist**
`403 Forbidden`	**Cookie** `JSESSIONID ljkhasioutfdqbjsfdmakhflia` **already exist**
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`403 Forbidden`	`json parsing error`
`500 Internal Server Error`	`unknown error`

Clear allow list (DELETE)

Request

`DELETE`	`/v4/ase/firewall/whitelist?tag=all`

Copyright © 2025 Ping Identity Corporation

`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Response

Status code	Response body
`200 OK`	`Whitelist cleared`
`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`500 Internal Server Error`	`unknown error`

View allow list (POST)

Request

`GET`	`/v4/ase/firewall/whitelist`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

Response

Status code	Response body

`200 OK`	<pre>{
	"whitelist" : [
	{
	"type" : "cookie",
	"name" : "JSESSIONID",
	"value" : "ljkhasiosalia",
	},
	{
	"type" : "ip",
	"value" : "1.1.1.1",
	}
	]
	}</pre>

`403 Forbidden`	`content-type header missing`
`403 Forbidden`	`x-ase-access-key header missing`
`403 Forbidden`	`x-ase-secret-key header missing`
`403 Forbidden`	`authorization failure`
`500 Internal Server Error`	`unknown error`

Read flow control of an API (GET)

Request

`GET`	`/v4/ase/firewall/flowcontrol?api_id=<api_name>`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

Copyright © 2025 Ping Identity Corporation

Response

HTTP Code	Status	Content body (application/JSON)

`200`	`success`	**Flow control for REST API:**
		`{ "api_id": "api_name" "flow_control": { "client_spike_threshold": "0/second", "server_connection_queueing": false } }`
		Flow control for WebSocket API:
		`{ "api_id": "api_name" "flow_control": { "client_spike_threshold": "100/second", "bytes_in_threshold": "10/second", "bytes_out_threshold": "10/second", "server_connection_queueing": false } }`

`403`	`fail`	`{"status" : "validation_error" , "status_message" : "<detailed validation error description" }`
`404`	`fail`	`{"status" :"api_not_found" ,"status_message" :"api sample does not exist"}`

Update flow control for an API (POST)

Request

`POST`	`/v4/ase/firewall/flowcontrol?api_id=<api_name>`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

REST APIs:

Copyright © 2025 Ping Identity Corporation

```
{ "flow_control": {
 "client_spike_threshold": "0/second"
 }
 }
```

WebSocket APIs

```
{ "flow_control": {
 "client_spike_threshold": "10/second",
 "bytes_in_threshold": "10/second",
 "bytes_out_threshold": "10/second"
 }
}
```

Response

HTTP Code	Status	Content body (application/JSON)

`200`	`success`	**Flow control for REST APIs:**
		`{ "api_id": "api_name" "flow_control": { "client_spike_threshold": "0/second", "server_connection_queueing": false } }`
		Flow control for WebSocket APIs:
		`{ "api_id": "api_name" "flow_control": { "client_spike_threshold": "0/second", "bytes_in_threshold": "10/second", "bytes_out_threshold": "10/second", "server_connection_queueing": false }}`

`403`	`fail`	`{"status" : "validation_error" , "status_message" : "<detailed validation error description" }`
`404`	`fail`	`{"status" :"api_not_found" ,"status_message" :"api sample does not exist"}`

Copyright © 2025 Ping Identity Corporation

Update flow control for a server of an API (POST)

Request

`POST`	`/v4/ase/firewall/flowcontrol/server?api_id=<api_name>`
`x-ase-access-key`	`<Access Key>`
`x-ase-secret-key`	`<Secret Key>`
`Accept`	`application/json`

REST APIs:

```
{
 "server":
 {
 "host": "127.0.0.2",
 "port": 8080,
 "server_connection_quota": 1000,
 "server_spike_threshold": "10/second"
 }
}
```

WebSocket APIs:

```
{
 "server":
 {
 "host": "127.0.0.2",
 "port": 8080,
 "server_connection_quota": 100000
 }
}
```

Response

HTTP Code	Status	Content body (application/JSON)
`200`	`success`	`{ "status": "success", "status_message": "server updated successfully" }`

Copyright © 2025 Ping Identity Corporation

HTTP Code	Status	Content body (application/JSON)
`403`	`fail`	`{“status” : “validation_error” , “status_message” : “<detailed validation error description” }`
`404`	`fail`	`{“status” :”api_not_found” ,”status_message” :”api sample does not exist”}`

Audit log

This appendix details audit log entries in the `audit.log` file.

The following table shows the four components of entries in the audit log files.

Date	Subject	Action	Resources
`YYYY-MM-DD hh:mm:ss`	Subject is the module through which actions are performed: command-line interface (CLI), REST application programming interface (API), or cluster	Actions are the ran commands.	Resources are the parameters associated with the actions.

The following table shows the subjects and their description.

Subject	Description
`cli`	CLI commands ran
`rest_api`	REST API requests received by API Security Enforcer (ASE)
`cluster`	Changes requested by peer node in a cluster

Here is sample output of an audit log file:

```
2019-06-13 10:45:12 | cli | delete_api | username=admin, api_id=cart
2019-06-13 10:46:13 | rest_api | GET /v4/ase/cluster | x-ase-access-key=admin, x-ase-secret-key=
2019-06-13 10:46:25 | cluster | delete_api | peer_node=192.168.11.108:8020, api_id=shop
```

CLI

The following table lists the actions and resources for ASE CLI.

Copyright © 2025 Ping Identity Corporation

Action	Resources
`status`	N/A
`add_api`	`username=, config_file_path=`
`list_api`	`username=`
`api_info`	`username=, api_id=`
`api_count`	`username=`
`list_api_mappings`	`username=`
`delete_api`	`username=, api_id=`
`add_server`	`username=, api_id=, server=, server_spike_threshold=, server_connection_quota=`
`list_server`	`username=, api_id=`
`server_count`	`username=, api_id=`
`delete_server`	`username=, api_id=, server=`
`create_key_pair`	`username=`
`create_csr`	`username=`
`create_self_sign_cert`	`username=`
`import_cert`	`username=, cert_path=`
`health_status`	`username=, api_id=`
`enable_health_check`	`username=, api_id=`
`disable_health_check`	`username=, api_id=`
`update_password`	`username=`
`cluster_info`	`username=`
`cookie_count`	`username=, api_id=`
`enable_firewall`	`username=`
`disable_firewall`	`username=`
`enable_abs`	`username=`

Copyright © 2025 Ping Identity Corporation

Action	Resources
`disable_abs`	`username=`
`enable_abs_attack`	`username=`
`disable_abs_attack`	`username=`
`abs_info`	`username=`
`enable_xff`	`username=`
`disable_xff`	`username=`
`update_bytes_in_threshold`	`username=, api_id=, bytes_in_threshold=`
`update_bytes_out_threshold`	`username=, api_id=, bytes_out_threshold=`
`update_client_spike_threshold`	`username=, api_id=, client_spike_threshold=`
`update_server_spike_threshold`	`username=, api_id=, server=, server_spike_threshold=`
`update_server_connection_quota`	`username=, api_id=, server=, server_connection_quota`
`get_auth_method`	N/A
`update_auth_method`	`username=, auth_method=`
`enable_audit`	`username=`
`disable_audit`	`username=`
`stop`	`username=`

REST API

Action	Resource
`POST /v4/ase/api`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`GET /v4/ase/api`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`DELETE /v4/ase/api`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`POST /v4/ase/server`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`

Copyright © 2025 Ping Identity Corporation

Action	Resource
`GET /v4/ase/server`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`DELETE /v4/ase/server`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`GET /v4/ase/cluster`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`POST /v4/ase/firewall`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`GET /v4/ase/firewall`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`POST /v4/ase/firewall/flowcontrol`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`GET /v4/ase/firewall/flowcontrol`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`
`POST /v4/ase/firewall/flowcontrol/server`	`Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=`

Cluster

Action	Resource
`add_api`	`peer_node=, api_id=`
`delete_api`	`peer_node=, api_id=`
`add_server`	`peer_node=, api_id=, server=, server_spike_threshold=, server_connection_quota=`
`delete_server`	`peer_node=, api_id=, server`
`enable_health_check`	`peer_node=, api_id=`
`disable_health_check`	`peer_node=, api_id=`
`enable_firewall`	`peer_node=`
`disable_firewall`	`peer_node=`
`enable_abs`	`peer_node=`

Copyright © 2025 Ping Identity Corporation

Action	Resource
`disable_abs`	`peer_node=`
`enable_abs_attack`	`peer_node=`
`disable_abs_attack`	`peer_node=`
`enable_xff`	`peer_node=`
`disable_xff`	`peer_node=`
`update_bytes_in_threshold`	`peer_node=, api_id=, bytes_in_threshold=`
`update_bytes_out_threshold`	`peer_node=, api_id=, bytes_out_threshold=`
`update_client_spike_threshold`	`peer_node=, api_id=, client_spike_threshold=`
`update_server_spike_threshold`	`peer_node=, api_id=, server=, server_spike_threshold=`
`update_server_connection_quota`	`peer_node=, api_id=, api_id=, server=, server_connection_quota=`
`enable_audit`	`peer_node=`
`disable_audit`	`peer_node=`
`stop`	`peer_node=`

Supported encryption protocols

The following tables show a complete list of supported encryption protocols for TLS1.2 based on the operating system.

RHEL 7.6

`ECDHE-RSA-AES256-GCM-SHA384`	`ECDHE-ECDSA-AES128-GCM-SHA256`
`ECDHE-ECDSA-AES256-GCM-SHA384`	`DH-RSA-AES128-GCM-SHA256`
`ECDHE-RSA-AES256-SHA384`	`ECDHE-RSA-AES128-SHA256`
`ECDHE-ECDSA-AES256-SHA384`	`ECDHE-ECDSA-AES128-SHA256`
`DHE-DSS-AES256-GCM-SHA384`	`DHE-DSS-AES128-GCM-SHA256`
`DHE-RSA-AES256-GCM-SHA384`	`DHE-RSA-AES128-GCM-SHA256`
`DHE-RSA-AES256-SHA256`	`DHE-RSA-AES128-SHA256`

Copyright © 2025 Ping Identity Corporation

DHE-DSS-AES256-SHA256	DHE-DSS-AES128-SHA256
ECDH-RSA-AES256-GCM-SHA384	ECDH-RSA-AES128-GCM-SHA256
ECDH-ECDSA-AES256-GCM-SHA384	ECDH-ECDSA-AES128-GCM-SHA256
ECDH-RSA-AES256-SHA384	ECDH-RSA-AES128-SHA256
ECDH-ECDSA-AES256-SHA384	ECDH-ECDSA-AES128-SHA256
AES256-GCM-SHA384	AES128-GCM-SHA256
AES256-SHA256	AES128-SHA256
ECDHE-RSA-AES128-GCM-SHA256	

Ubuntu 16.04

ECDHE-RSA-AES256-GCM-SHA384	DHE-DSS-AES128-GCM-SHA256
ECDHE-ECDSA-AES256-GCM-SHA384	DHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES256-SHA384	DHE-RSA-AES128-SHA256
ECDHE-ECDSA-AES256-SHA384	DHE-DSS-AES128-SHA256
DHE-DSS-AES256-GCM-SHA384	ECDH-RSA-AES128-GCM-SHA256
DHE-RSA-AES256-GCM-SHA384	ECDH-ECDSA-AES128-GCM-SHA256
DHE-RSA-AES256-SHA256	ECDH-RSA-AES128-SHA256
DHE-DSS-AES256-SHA256	ECDH-ECDSA-AES128-SHA256
ECDH-RSA-AES256-GCM-SHA384	AES128-GCM-SHA256
ECDH-ECDSA-AES256-GCM-SHA384	AES128-SHA256
ECDH-RSA-AES256-SHA384	DH-RSA-AES128-GCM-SHA256
ECDH-ECDSA-AES256-SHA384	DH-DSS-AES128-GCM-SHA256
AES256-GCM-SHA384	DH-RSA-AES128-SHA256
AES256-SHA256	DH-DSS-AES128-SHA256
ECDHE-RSA-AES128-GCM-SHA256	DH-DSS-AES256-GCM-SHA384
ECDHE-ECDSA-AES128-GCM-SHA256	DH-RSA-AES256-GCM-SHA384

Copyright © 2025 Ping Identity Corporation

`ECDHE-RSA-AES128-SHA256`	`DH-RSA-AES256-SHA256`
`ECDHE-ECDSA-AES128-SHA256`	`DH-DSS-AES256-SHA256`

Autoscaling ASE in an AWS environment

You can auto-scale API Security Enforcer (ASE) setup in an Amazon Web Services (AWS) environment.

Steps

1. Create an Amazon Machine Image (AMI) for ASE.
2. Create an identity and access management (IAM) role in the security, identity, and compliance.
3. Create the security group.
4. Create launch configuration.
5. Create an auto-scale group.

Creating an AMI for ASE

Create an Amazon Machine Image (AMI) for API Security Enforcer (ASE).

Steps

1. Create an RHEL 7.6 or Ubuntu 16.04 LTS EC2 instance.
2. Install the Amazon Web Services (AWS) command-line interface (CLI):
    1. Install Python 2.7.
    2. Run the following command:

    ```
    sudo curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
    ```

    3. Extract the CLI bundle:

    ```
    sudo unzip awscli-bundle.zip
    ```

    4. Install the CLI:

    ```
    sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/bin/aws
    ```

3. Download the ASE AWS binary. After downloading the file, copy the ASE file to the `/opt` directory.
4. To untar the binary in the EC2 instance, at the command prompt, enter the following command to untar the ASE file:

Copyright © 2025 Ping Identity Corporation

```
tar –zxvf ase-rhel-4.0.tar.gz
```

Result:

```
tar –zxvf  <filename>
```

5. To verify that ASE successfully installed, enter the `ls` command at the command prompt:

```
/opt/$ ls
pingidentity ase-rhel-4.0.tar.gz
```

Result:

This lists the PingDirectoryand the build's tar files.

6. Change the directory to `/opt/pingidentity/ase/bin`.
7. Run the `install_service.sh aws` script:

```
/opt/pingidentity/ase/bin$sudo ./install_service.sh aws
Installing ASE service for AWS Autoscale
This script will install ASE as a service
Do you wish to proceed (y/n)? y
Starting service installation
RHEL7.6 detected, installing ASE service
Created symlink from /etc/systemd/system/multi-user.target.wants/ase.service to /etc/systemd/system/
ase.service.
ASE service successfully installed
```

8. Create an AMI using this EC2 instance.

> **Important**
>
> When you are creating the AMI, do not select the `No Reboot` option.

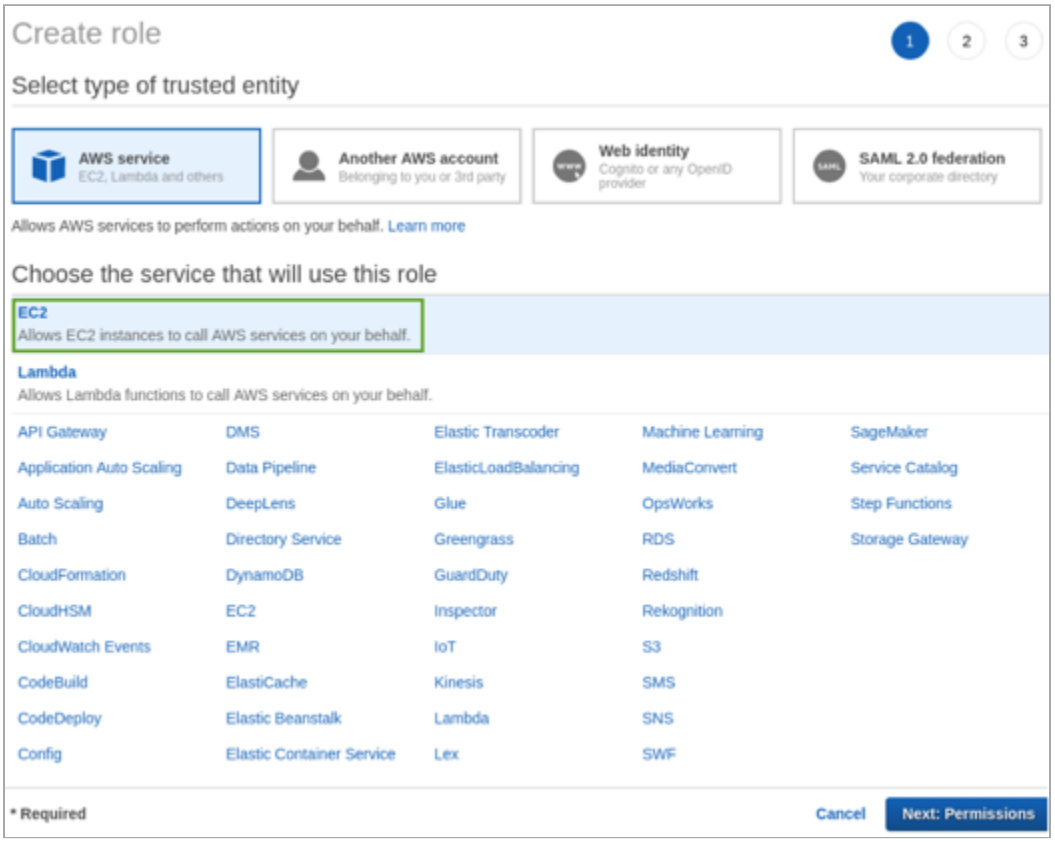
Creating an IAM role in the security, identity, and compliance

Create an identity and access management (IAM) role in the security, identity, and compliance.

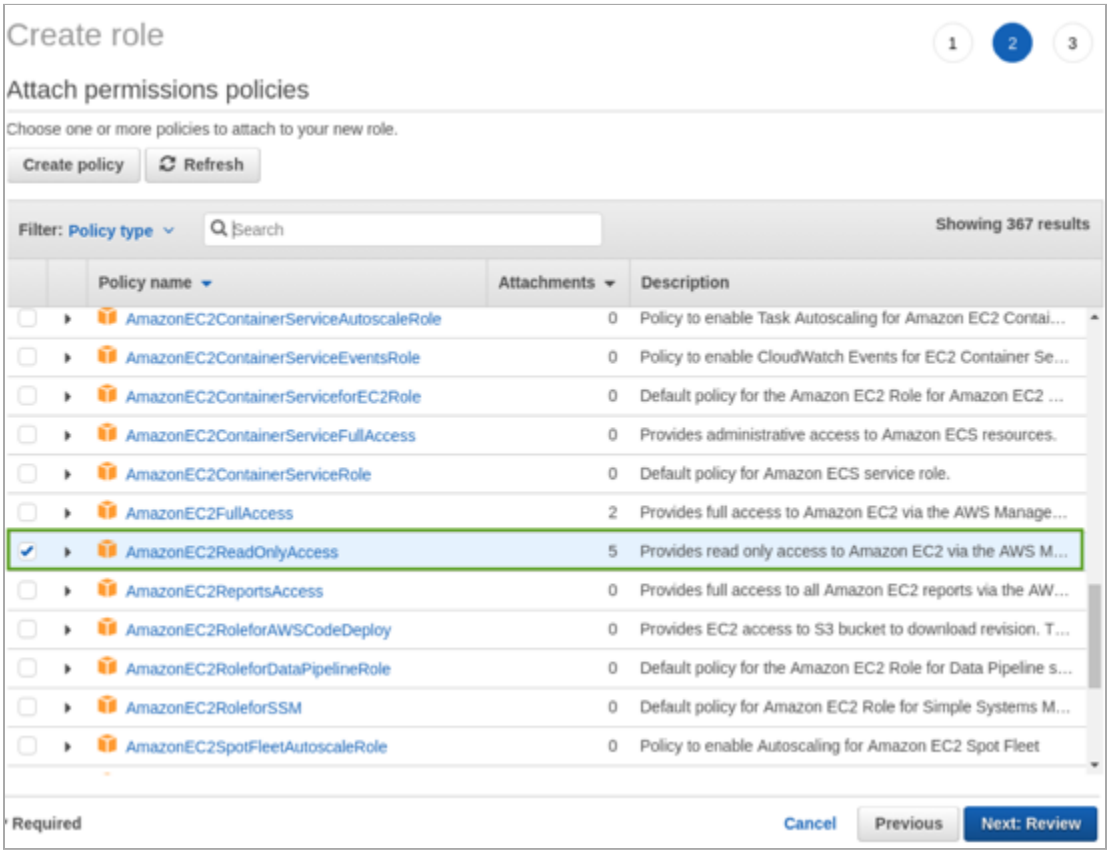
Steps

1. To create an IAM role, select the EC2 instance.

Copyright © 2025 Ping Identity Corporation

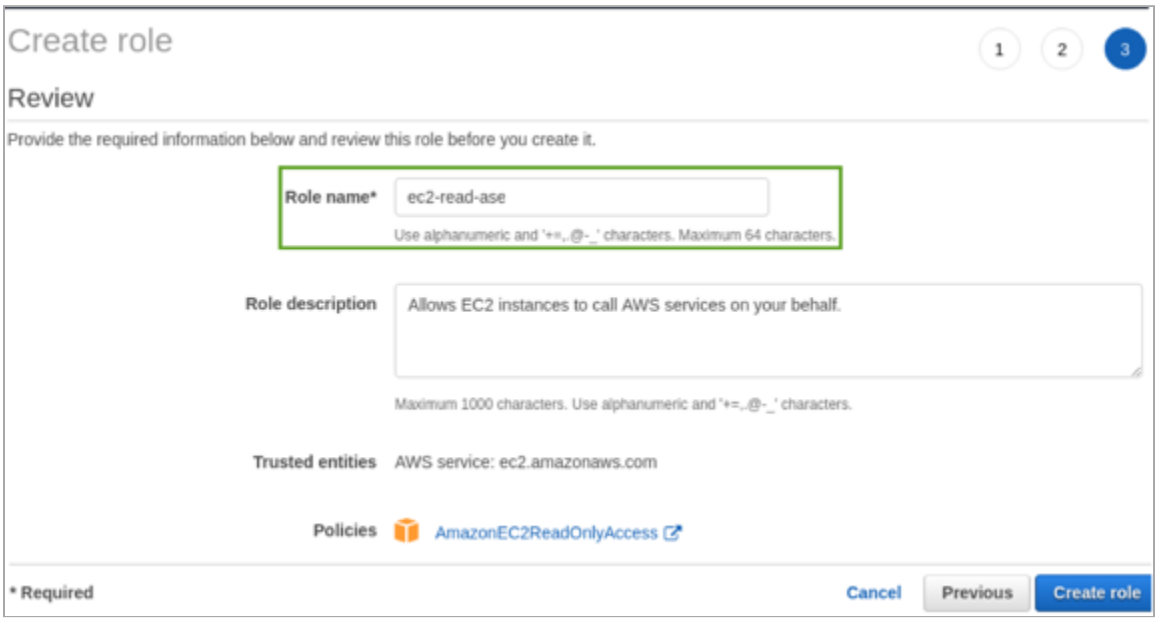


2. Assign the AmazonEC2ReadOnlyAccess privilege to the role.



Copyright © 2025 Ping Identity Corporation

3. Enter the role name.



Create the security group

You must create a security group for the following ports used by the API Security Enforcer (ASE):

- Port `80`: Accessible by API Clients/ELB
- Port `443`: Accessible by API Clients/ELB
- Port `8010`: Accessible by operations to execute command-line interface (CLI) commands and REST application programming interface (API) calls
- Port `8020`: Only accessible by peer ASE nodes in the same security group

Create a security group based on the following table.

Type	Protocol	Port	Source
Custom TCP	TCP	`80`	API clients/ELB
Custom TCP	TCP	`443`	Same security group
Custom TCP	TCP	`8010`	Same security group
Custom TCP	TCP	`8020`	Same security group

Creating launch configuration

You must create the launch configuration that the auto-scaling group will use.

Copyright © 2025 Ping Identity Corporation

Steps

1. Select the Amazon Machine Image (AMI) that you created in Create an AMI for ASE.
2. Create the EC2 instance based on the sizing requirement.
3. Assign the identity and access management (IAM) role that you created in Create an IAM Role in the Security, Identity, and Compliance to the launch configuration.
4. Complete the creation of the launch configuration.

Creating an auto-scale group

Create the auto-scale group.

Steps

1. Use the launch configuration that created in Creating launch configuration.
2. Optional: Attach the ELB to the auto-scale group that you created in step 1.
3. Configure the following rules for the auto scale group:
    1. To configure the Increase Group Size rule, add one instance when the Average CPU utilization is greater than 90% for at least 2 consecutive periods of 5-minutes.
    2. To configure the Decrease Group Size rule, remove one instance when the Average CPU utilization is less than 10% for at least two consecutive periods of 5-minutes.

Next steps

Optionally, you can uninstall the API Security Enforcer (ASE) service installed in Creating an AMI for ASE by running the `uninstall_service.sh` command, which results in an output similar to the following:

```
/opt/pingidentity/ase/bin$sudo ./uninstall_service.sh
This script will uninstall ASE service
Do you wish to proceed (y/n)? y
Starting service uninstallation
RHEL 7.6 detected, uninstalling ASE Service..
ase stop/waiting
ASE service successfully uninstalled
```

ASE log messages

The following tables list the critical log messages from `controller.log` and `balancer.log` files. Note that the `balancer.log` file is not rotated while the `controller.log` file is rotated every 24 hours. For more information on API Security Enforcer (ASE) logs, see ASE access, management, and audit logs.

Copyright © 2025 Ping Identity Corporation

`controller.log` *messages:*

Log message	Description
unknown cluster uuid	This message is logged in `controller.log` when an ASE node with a different cluster ID or secret key tries to join an ASE cluster. For more information, see Setting up an ASE cluster.
resolve error	This message is logged in `controller.log` when ASE is not able to resolve API Behavioral Security (ABS) or server hostname.
connect error	This message is logged in `controller.log` when ASE is not able to connect to ABS or a server.
handshake error	This message is logged in `controller.log` when connection to ABS or server because of problems in SSL handshake.
error while sending message to lb connection	This message is logged in `controller.log` when there is a IPC connection failure between ASE's controller and balancer modules.
error while reading message from lb connection	This message is logged in `controller.log` when there is a IPC connection failure between ASE's controller and balancer modules.
License file <*license file path*> is expired. Please renew your license	This message is logged in `controller.log` when PingIntelligence license has expired. For more information, see Configuring and updating an ASE license.
Unexpected Error	This message is logged in `controller.log` when ASE's controller module is unavailable. This is a fatal error.
info | event | event type: `<event type>` event value : <*value of event*>	The following events are logged logs even if email alert is not enabled: • Cluster node up • Cluster node down • server state changed to Up • server state changed to Down • log upload service failed • error while uploading log file If email_alert is enabled, then all events will be available in logs. For more information, see Email alerts and reports.

Copyright © 2025 Ping Identity Corporation

Log message	Description
api memory limit reached. total number of cookies dropped <*count*>	This message is logged in `controller.log` when ASE is dropping cookies because of low API memory. For more information, see `api_memory_size` in Defining an API using API JSON configuration file in sideband mode.
stopping API Security Enforcer	This message is logged in `controller.log` when ASE stops.
API Security Enforcer started	This message is logged in `controller.log` when ASE starts.

`balancer.log` *messages*

Log message	Description
`/bin/bash` : line 0: echo: write error: Permission denied `/bin/bash` : line 0: echo: write error: Permission denied	
| warn | process_id : process_id_number | tcp_fe | !!!! low memory : connection dropped due to low memory !!!!	This message is logged in `controller.log` when ASE is running low on memory because of which ASE drops the client connections.

ABS AI Engine

The API Behavioral Security (ABS) AI engine is a Java-based distributed system that analyzes application programming interface (API) traffic to provide API traffic insight, visibility, and security.

API traffic information is received from API Security Enforcer (ASE) nodes in log files containing:

- Client details such as device, browser, Internet Protocol (IP) address, and operating system.
- Session information including HTTP or WebSocket connections and methods.

These logs are periodically (every 10 minutes) forwarded to ABS nodes for processing. Using machine learning algorithms, ABS generates API traffic insight, anomaly data, and attack insight that identifies clients responsible for attacks. To prevent future attacks, ABS can automatically program inline devices, such as the ASE , to block clients based on attack lists.

The ABS AI engine provides the following functionality:

- Collection and consolidation of access logs from ASE nodes
- Machine learning algorithms to identify anomalies and attacks
- Detection of attacks from HTTP(s) and WebSocket(s) traffic
- Optional sending of blacklists to ASE which blocks client access
- Centralized database for storing AI data
- Stateless cluster for scalability and resiliency

Copyright © 2025 Ping Identity Corporation

- REST APIs for fetching traffic metrics, anomalies, and attack information
- Email alerts

Configuring ABS consists of setting up two entities:

Database system

ABS uses a MongoDB database to store metadata and all Machine Learning (ML) analytics. The MongoDB database system is configured in a replica set for production deployments. MongoDB is separately installed before starting ABS.

ABS AI engine

One or more ABS instances are configured to receive and process logs and to store results in MongoDB. You should install ABS in a cluster for high availability deployments.

ABS Administration

Administering API Behavioral Security (ABS) requires the understanding of:

- Directory structure
- Obfuscating passwords for securing ABS
- Configuring Secure Sockets Layer (SSL) for secure communication for between PingIntelligence products
- Different types of ABS users
- Understanding the port requirements
- Creating ABS cluster
- Understanding ABS log files
- Purging access logs from ABS
- ABS REST API format

Configuring and updating an ABS license

To start API Behavioral Security (ABS), you need a valid license.

There are two types of ABS licenses:

Trial license

The trial license is valid for 30 days. At the end of the trial period, ABS stops processing and shuts down.

Subscription license

The subscription license is based on the subscription period.

Copyright © 2025 Ping Identity Corporation

> **Important**
>
> You should configure your email before configuring the ABS license. ABS sends an email notification to the configured email ID when the license has expired.

For more information, contact the PingIntelligence for APIssales team.

> **Note**
>
> If the subscription license has expired, ABS continues to run until a restart. ABS needs a valid license file to start.

Checking the current transaction count

Use the Admin REST API to view the current transaction count against your subscribed transaction limit. Following snippet of the Admin REST API shows the license information:

```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, and ASE logs",
    "license_info": {
        "tier": "Subscription",
        "expiry": "Wed Jan 15 00:00:00 UTC 2020",
        "max_transactions_per_month": 1000000000,
        "current_month_transactions": 98723545,
        "max_transactions_exceeded": false,
        "expired": false
    }
```

Configuring ABS license

About this task

To configure the API Behavioral Security (ABS) license:

Steps

1. **Request a license file from the PingIntelligence for APIs sales team.**

   > **Note**
   >
   > The name of the license file must be `PingIntelligence.lic`

2. **Copy the license file to the `/opt/pingidentity/abs/config` directory**
3. **Start ABS.**

Updating an existing ABS license

Before you begin

If your existing license has expired, obtain a new license from the PingIntelligence for APIs sales team.

About this task

Copyright © 2025 Ping Identity Corporation

To update an existing API Behavioral Security (ABS) license:

Steps

1. Replace the license file in the `/opt/pingidentity/abs/config` directory
2. After you have updated the license file, stop and then start ABS after the license file is updated.

Changing default settings

For security reasons, you should change the default master key and passwords in API Behavioral Security (ABS).

Before you begin

> **Note**
> Make sure that ABS is stopped before changing the keystore password.

About this task

To change the default values:

Steps

- To change the keystore password, enter the following command.

  The default Java KeyStore (JKS) password is `abs123`.

  ```
  # keytool -storepasswd -keystore config/ssl/abs.jks
  Enter keystore password:  abs123
  New keystore password: newjkspassword
  Re-enter new keystore password: newjkspassword
  ```

- To change the key password, enter the following command.

  The default key password is `abs123`.

  ```
  # keytool -keypasswd -alias pingidentity -keypass abs123 -new newjkspassword -keystore config/ssl/
  abs.jks
  Enter keystore password: newjkspassword
  ```

  > **Note**
  > Start ABS after you have changed the default passwords.

- Before creating a new `abs_master.key`, stop ABS by running the `stop.sh` command.

Copyright © 2025 Ping Identity Corporation

```
# /opt/pingidentity/abs/bin/stop.sh
checking API Behavioral Security status
sending shutdown signal to ABS, please wait...
API Behavioral Security stopped
```

- To create your own `abs_master.key` to obfuscate keys and passwords in ABS, run the `generate_obfkey` command.

```
/opt/pingidentity/abs/bin/cli.sh generate_obfkey -u admin -p admin
Please take a backup of config/abs_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate all config keys also
using cli.sh -obfuscate_keys
Warning: Obfuscation master key file
/pingidentity/abs/config/abs_master.key already exists. This command will delete it and create a new
key in the same file
Do you want to proceed [y/n]: y
Creating new obfuscation master key
Success: created new obfuscation master key at /pingidentity/abs/config/abs_master.key
```

- To change the default admin password, run the `update_password` command.

```
/opt/pingidentity/abs/bin/cli.sh update_password -u admin -p admin
New Password>
Reenter New Password>
Success. Password updated for CLI
```

- To change the default access and secret key in MongoDB, stop the ABS nodes and complete the following:

  1. Connect to MongoDB by entering the following command.

     `absuser` and `abs123` are the default username and password for MongoDB.

     ```
     mongo --host<mongo-host>--port  <mongo-port>--authenticationDatabase admin -u absuser -p
     abs123
     ```

  2. On the MongoDB prompt, run the following command:

     ```
     use abs_metadata
     db.auth_info.updateOne( { access_key: "<new-access-key>", secret_key: "<new-secret-key>"} )
     ```

  3. Start the ABS nodes after you have changed the default access and secret key.

Obfuscating keys and passwords

Using API Behavioral Security (ABS) command line interface, you can obfuscate the keys and passwords configured in `abs.properties`.

About this task

Copyright © 2025 Ping Identity Corporation

The keys and passwords obfuscated include:

- `mongo_password`
- `jks_password`
- `email_password`

ABS ships with a default `abs_master.key` which is used to obfuscate the keys and passwords. It is recommended to generate your own `abs_master.key`.

The following diagram summarizes the obfuscation process.

Steps

1. To obfuscate keys and passwords, stop ABS.
2. To generate your `abs_master.key`, run the `generate_obfkey` ABS CLI command.

```
/opt/pingidentity/abs/bin/cli.sh generate_obfkey -u admin -p admin
Please take a backup of config/abs_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate all config keys also
using cli.sh -obfuscate_keys
Warning: Obfuscation master key file
/pingidentity/abs/config/abs_master.key already exists. This command will delete it and create a new
key in the same file
Do you want to proceed [y/n]: y
Creating new obfuscation master key
Success: created new obfuscation master key at /pingidentity/abs/config/abs_master.key
```

   The new `abs_master.key` is used to obfuscate the passwords in `abs.properties` file.

   > **Important**
   >
   > After the keys and passwords are obfuscated, the `abs_master.key` must be moved to a secure location and not stored on ABS.
   > In an ABS cluster, the `abs_master.key` must be manually copied to each of the cluster nodes.

3. To obfuscate key and passwords, enter the keys and passwords in clear text in the `abs.properties` file.
4. Run the `obfuscate_keys` command.

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/abs/bin/cli.sh obfuscate_keys -u admin -p admin
Please take a backup of config/abs.password before proceeding
Enter clear text keys and passwords before obfuscation.
Following keys will be obfuscated
config/abs.properties: mongo_password, jks_password and email_password
Do you want to proceed [y/n]: y
obfuscating /pingidentity/abs/config/abs.properties
Success: secret keys in /pingidentity/abs/config/abs.properties obfuscated
```

5. Start ABS after passwords are obfuscated.

Configuring and verifying ABS POC mode

You can run API Behavioral Security (ABS) AI engine in proof of concept (POC) mode which requires substantially lesser number of requests for detecting an attack.

This mode is only for the purpose of demonstrating the capabilities of the AI engine. All the REST API attacks can be detected in the POC mode. For more information on different attack types, see Attack Types REST and WebSocket APIs.

> **Warning**
>
> Do not deploy the AI engine in production environment in POC mode. It is recommended to uninstall all PingIntelligence components from POC mode and reinstall for production environment. The ABS AI engine in POC mode is not suitable for security testing as well.

Configuring ABS POC mode

About this task

You can install API Behavioral Security (ABS) AI engine in POC mode.

Steps

- Configure the parameter during automated installation. For more information on configuring the POC mode at the time of installation, see Changing ABS default settings.
- If you are using manual installation to install ABS AI engine and MongoDB, configure `poc` to `true` using the Global configuration update REST API.

Verifying ABS POC mode

About this task

Use the API Behavioral Security (ABS) Admin REST API to verify whether ABS AI engine is running in the POC mode.

Steps

- You can access the report by calling the ABS application programming interface (API) at https://*<abs_ip>*:*<abs_port>*/v5/abs/admin.

Copyright © 2025 Ping Identity Corporation

Example:

The following example shows where `poc` has been set to `true` in bold:

```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, and ASE logs",
    "license_info": {
        "tier": "Subscription",
        "expiry": "Sun Feb 21 00:00:00 UTC 2021",
        "max_transactions_per_month": 100000000,
        "current_month_transactions": 41243418,
        "max_transactions_exceeded": false,
        "expired": false
    },
    "across_api_prediction_mode": true,
     "poc": true,
    "api_discovery": {
        "subpath_length": "1",
        "status": true
    },

...truncated admin API output...
}
```

Starting and stopping ABS

You can start and stop API Behavioral Security (ABS).

Stop ABS before performing any of the following tasks:

- When deleting the ABS directory.
- When deleting the data or metadata database.
- When changing the user permissions.

Failure to do so will result in excessive logs in the MongoDB node.

Copyright © 2025 Ping Identity Corporation

Starting ABS

Starting ABS
Steps

1. **To start API Behavioral Security (ABS), run the `start.sh` script located in the `/opt/pingidentity/abs/bin` directory.**

   > **Note**
   >
   > To start ABS, you must accept the EULA. You can accept the EULA in two ways:
   > - Scroll through the text on screen and enter `yes` to accept the EULA.
   > - Use the `--acceptLicense` option with `start.sh` as shown in the screen output below:
   >
   > ```
   > $ /opt/pingidentity/abs/bin/start.sh --acceptLicense
   > End-User License Agreement accepted
   > Starting API Behavioral Security Version 4.1...
   > please see /opt/pingidentity/abs/logs/abs/abs.log for more details
   > ```
   >
   > After you accept the EULA, ABS creates a `license.accepted` file in the `/opt/pingidentity/abs/config` directory, which ABS checks for on subsequent starts.

2. **To verify that ABS has started, change the working directory to the `data` directory and look for two `.pid` files, `abs.pid` and `stream.pid`.**

3. **Make sure that the newly added ABS node is connecting to MongoDB and has a heartbeat.**

   Example:

```
> use abs_metadata
switched to db abs_metadata
> db.abs_cluster_info.find().pretty()
 {
 "_id" : ObjectId("58d0c633d78b0f6a26c056ed"),
 "cluster_id" : "c1",
 "nodes" : [
 {
 "os" : "Red Hat Enterprise Linux Server release 7.1 (Maipo)",
 "last_updated_at" : "1490088336493",
 "management_port" : "8080",
 "log_port" : "9090",
 "cpu" : "24",
 "start_time" : "1490077235426",
 "log_ip" : "2.2.2.2",
 "uuid" : "8a0e4d4b-3a8f-4df1-bd6d-3aec9b9c25c1",
 "dashboard_node" : false,
 "memory" : "62G",
 "filesystem" : "28%"
 } ] }
```


Copyright © 2025 Ping Identity Corporation

Stopping ABS

Stopping ABS
Steps

1. Stop API Security Enforcer (ASE) or turn off the ABS flag in ASE.
2. If no machine learning jobs are processing, run the `stop.sh` script available in the `bin` directory.

   Example:

   ```
   # /opt/pingidentity/abs/bin/stop.sh
   checking API Behavioral Security status
   sending shutdown signal to ABS, please wait...
   API Behavioral Security stopped
   ```

   Note

   If streaming or machines learning jobs are in progress, add the `force` parameter to kill running jobs and stop ABS:

   ```
   # /opt/pingidentity/abs/bin/stop.sh  --force
   checking API Behavioral Security status
   sending shutdown signal to ABS, please wait...
   API Behavioral Security stopped
   ```

ABS users for API reports

API Behavioral Security (ABS) has two types of users to access the application programming interface (API) reports and PingIntelligence Dashboard.

The API reports displayed is based on the type of user accessing the reports. The two users are:

- Admin user: An Admin user has complete access to API reports. All the cookies, tokens, API keys, and Username are visible in the reports. Use the following headers in the API report URL to access API reports as an Admin user:
  - `x-abs-ak (access key header)`
  - `x-abs-sk (secret key header)`
- Restricted user: A Restricted user has limited access to the API reports. The Restricted user can view the API reports however the cookies, tokens, and API keys are obfuscated. Use the following headers in the API report URL to access API reports as an Admin user:
  - `x-abs-ak-ru (access key header)`
  - `x-abs-sk-ru (secret key header)`

  The restricted user can access all the API Reports except:
  - Threshold API

Copyright © 2025 Ping Identity Corporation

- Cookie, OAuth2 Token, Internet Protocol (IP), API Key, and Username Forensics APIs

For a complete list of external REST APIs, see ABS External REST APIs.

The default access and secret key are configured in the `opt/pingidentity/mongo/abs_init.js` file. Following is a snippet of the `abs_init.js` showing the default passwords for both type of users.

```
db.auth_info.insert({
 "access_key": "abs_ak",
 "secret_key": "abs_sk",
 "access_key_ru" : "abs_ak_ru",
 "secret_key_ru" : "abs_sk_ru"
});
```

> **Note**
>
> You can use **`update_keys`** CLI command to change access and secret keys. For more information, see ABS CLI commands.

ABS directory structure

API Behavioral Security (ABS) creates directories as part of the installation process.

The directories are shown in the following table.

Directory	Purpose
`config`	Contains `abs.properties`, a Java properties file used to configure ABS.
`data`	Stores logs sent by application programming interface (API) Security Enforcer.
`logs`	Stores all ABS related logs.
`lib`	For internal use. Do not change anything in this directory.
`bin`	Contains various scripts to start and stop ABS. **Note** Do not edit the scripts in the `bin` directory.
`mongo`	Contains the `abs_init.js` file used to load the default schema, secret key, and access key.

Copyright © 2025 Ping Identity Corporation

Directory	Purpose
`util`	Contains utilities to: • Check and Open MongoDB Default Port • Purge the Processed Access Logs from ABS • Purge ABS Data from MongoDB • Various service and systemctl scripts • Reset MongoDB script, and • Update script to change the values of global configuration defined in `/pingidentity/abs/mongo/abs_init.js` file • `open_ports_abs.sh:` Open the default ports 8080 and 9090 for ABS REST API and connectivity from ASE respectively. Run the script on the ABS machine.

Configure SSL

API Behavioral Security (ABS) supports only TLS 1.1 and TLS 1.2 and requires open Java Development Kit (JDK) 11.0.2.

ABS ships with a default self-signed certificate with Java Keystore at `abs/config/ssl/abs.jks` and the default password set to `abs123` in the `abs.properties` file. The default password is obfuscated in the `abs.properties` file. It is recommended to change the default passwords and obfuscate the new passwords. For more information, see Obfuscating keys and passwords.

If you want to use your own certificate authority (CA) signed certificates, you can import them in ABS.

> **Note**
>
> The Secure Sockets Layer (SSL) communication between API Security Enforcer (ASE) and ABS is by default enabled. If you need to disable it, contact Ping Identity support team.

Configuring the time zone in ABS

You can configure time zone API Behavioral Security (ABS).

About this task

To configure PingIntelligence, complete the following steps:

Steps

- Configure the `timezone` parameter in `/pingidentity/abs/config/abs.properties` file to set up ABS AI Engine in either `local` or `UTC` time zone.

  > **Note**
  >
  > If the `timezone` parameter is left empty, ABS runs in the `UTC` time zone by default.

  Example:

Copyright © 2025 Ping Identity Corporation

The following is a snippet of the `abs.properties` file for the `timezone` parameter.

```
# Set the timezone to utc or local. The default timezone is utc.
timezone=utc  <truncated abs.properties...>
```

> **Important**
>
> Make sure that API Security Enforcer (ASE), ABS AI Engine, and PingIntelligence are all configured in the same time zone.

Choose from:

- If ABS AI Engine is deployed in a cluster, make sure to configure the same time zone on each cluster node.
- If you have used manual installation, then you need to manually configure the time zone on each ABS node.

> **Note**
>
> You can use the Admin REST API to check the current time zone setting of ABS AI Engine. If you have used automated deployment to deploy PingIntelligence, the automated deployment configures the same time zone on each ABS node.

- To change the time zone in ABS AI Engine:

1. Stop ABS AI Engine.
2. Update the `timezone` parameter in the `abs.propeties` file.
3. Start ABS AI Engine.

> **Note**
>
> For more information, see Starting and stopping ABS.

Next steps

For more information, see ABS configuration - abs.properties.

Import existing CA-signed certificates

You can import your existing certificate authority certificate authority (CA) signed certificate in API Behavioral Security (ABS).

Before you begin

Stop ABS if it is already running to import the CA-signed certificate.

About this task

To import the CA-signed certificate:

Steps

1. Export your CA-signed certificate to the PKCS12 store by entering the following command:

Copyright © 2025 Ping Identity Corporation

```
# openssl pkcs12 -export -in  <your_CA_cerficate>.crt -inkey  <your_certificate_key>.key -out
abs.p12 -name  <alias_name>
```

Example:

```
# openssl pkcs12 -export -in ping.crt -inkey ping.key -out abs.p12 -name exampleCAcertificate
Enter Export Password:
Verifying - Enter Export Password:
```

> **Note**
>
> If you have an intermediate certificate from the CA, then append the content to the `<your_CA_certificate>.crt` file.

2. Import the certificate and key from the PKCS12 store to Java KeyStore (JKS) by entering the following command:

```
# keytool -importkeystore -destkeystore abs.jks -srckeystore abs.p12 -srcstoretype PKCS12 -alias
<alias_name>-storetype jks
```

Example:

```
# keytool -importkeystore -destkeystore abs.jks -srckeystore abs.p12 -srcstoretype PKCS12 -alias
exampleCAcertificate  -storetype jks
Importing keystore abs.p12 to abs.jks...
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
```

> **Note**
>
> The command requires the destination keystore password. The destination keystore password entered in the command should be the same as configured in the abs.properties file.

Example:

Here is a snippet of the `abs.properties` file where the destination keystore password is stored. The password is obfuscated.

```
# Java Keystore password
jks_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCulhAUKj7ww4=
```

3. Copy the `abs.jks` file that you created in step 2 to the `/opt/pingidentity/abs/config/ssl` directory.

4. Start ABS by entering the following command:

```
# /opt/pingidentity/abs/bin/start.sh
Starting API Behavioral Security 4.0...
please see /opt/pingidentity/abs/logs/abs/abs.log for more details
```

Copyright © 2025 Ping Identity Corporation

ABS ports

API Behavioral Security (ABS) uses ports 8080 and 27017. The table below shows a description of each port.

Port number	Description
8080	This port is used by API Security Enforcer (ASE) to log in to ABS and also used by Postman to access data to generate application programming interface (API) reports
27017	Default port for MongoDB

Check and open MongoDB default port

MongoDB's default port for connection with ABS is 27017. Run the `check_ports_abs.sh` script on the ABS machine to determine whether MongoDB's default port is available. Provide MongoDB host IP address and default port as arguments. For example: `/opt/pingidentity/abs/util/check_ports_abs.sh \{MongoDB IPv4:[port]}`

Check and open MongoDB default port

Run the `check_ports_abs.sh` script on the ABS machine to determine whether MongoDB's default port is available. Input the MongoDB host IP address and default port (27017) as arguments. For example:

```
/opt/pingidentity/util/check_ports_abs.sh \{MongoDB IPv4:[port]}
```

Run the script for MongoDB primary and secondary nodes. If the default ports are not accessible, open the port from the MongoDB machine.

ABS configuration - abs.properties

The API Behavioral Security (ABS) configuration file `abs.properties` is located in the ABS `config` directory.

The following table explains the parameters and provides recommended values.

Parameter	Description
ABS Internet Protocol (IP) , port, log level, and Java KeyStore (JKS) password	
`timezone`	Set the timezone to `utc` or `local`. The default timezone is `utc`.
`management_port`	Port for ABS to API Security Enforcer (ASE) and REST application programming interface (API) to ABS communication. The default value is 8080.
`log_level`	Log detail captured. The default is INFO. Additional options - DEBUG, ERROR, WARN, FATAL.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`jks_password`	The password of the JKS Keystore. ABS ships with a default obfuscated password. You can reset the password and obfuscate it. This password should be the same that you would use in importing your CA-signed certificate.
ABS performance configurations	
`system_memory`	Memory size in MB allocated to run machine learning jobs. Recommended to be at least 50% of system memory.
`queue_size`	Do not change the value of this parameter. The default is 10.
ABS email configurations for alerts and reporting	
`enable_emails`	Enable (`true`) or disable (`false`) ABS email notifications.
`sender_email`	Email address used for sending email alerts and reports.
`receiver_email`	Email address notified about alerts and reports. If you want more than one person to be notified, use an email alias.
`email_password`	Password of sender's email account. **Note** You can leave this field blank if your SMTP server does not require authentication.
`smtp_port`	Port number of SMTP server.
`smtp_host`	Hostname of SMTP server.
`smtp_ssl`	Set to `true` if you want email communication to be over SSL. Make sure that the SMTP server supports SSL. If you set `smtp_ssl` to `true` and the SMTP server does not support SSL, email communication falls back to the non-SSL channel. The default value is `true` . Set it to false if email communication is over a non-SSL channel. The email communication will fail if you set the parameter to `false` , but the SMTP server only supports SSL communication.

Copyright © 2025 Ping Identity Corporation

Parameter	Description

`smtp_cert_verification`	Set to `true` if you want ABS to verify the SMTP server's SSL certificate. The default value is `false`.
	If you set it to `false`, ASE does not verify SMTP server's SSL certificate; however, the communication is still over SSL.
	Note
	If you have configured an IP address as `smtp_host` and set `smtp_cert_verification` to `true`, then make sure that the certificate configured on the SMTP server has the following:
	`X509v3 extensions:`
	`X509v3 Key Usage:`
	`Key Encipherment, Data Encipherment`
	`X509v3 Extended Key Usage:`
	`TLS Web Server Authentication`
	`X509v3 Subject Alternative Name:`
	`IP Address: X.X.X.X`
	Here `x.x.x.x` is the IP address is the address configured in `smtp_host`.

MongoDB configurations	
`mongo_rs`	Comma separated MongoDB replica set URI. A maximum of three nodes can be configured.
`metadata_dbname`	The MongoDB metadata database name. The default value is `abs_metadata`.
`data_dbname`	The MongoDB data database name. The default value is `abs_data`.
`mldata_dbname`	The MongoDB machine learning database name. The default value is `abs_mldata`
`mongo_auth_mechanism`	Defines the method in which MongoDB authenticates. The possible values can be: • `NONE` - Set it to `NONE`, if authentication is not configured in MongoDB • `DEFAULT` - Set it to `DEFAULT`, if you want to use native MongoDB username and password. Prove the values in the next two variables. • `PLAIN` - Set it to `PLAIN`, if you want to use LDAP authentication. In this case, provide the LDAP username and password in the next two variables.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`mongo_username`	**Username of MongoDB.** **Note** Required for MongoDB authentication
`mongo_password`	**MongoDB password**
`mongo_ssl`	**Set it to `true` if MongoDB is configured to use SSL connections. The default value is `false`.**
`mongo_certificate`	**Set it to `true` if you want to verify MongoDB SSL server certificate when ABS connects to MongoDB. The default value is `false`.** **Note** Make sure `mongo_ssl` is set to `true` before setting `mongo_certificate` to `true`.
ABS reporting node	
`dashboard_node`	**When `true`, designated as a dedicated Reporting or Dashboard node. This ABS node does not process log data or participate in an ABS cluster.** **The default value is `false`.** **Note** Multiple nodes can be Reporting or Dashboard nodes.
Cloud and OAuth configurations **Important** The following parameters are applicable when ABS is running in `cloud` mode. These are preset configurations, which should not be edited.	
`bucket_name`	**The Amazon Web Services (AWS) S3 bucket name.**
`env_id`	**The environment id of the tenant.**
`deployment_type`	**The ABS deployment mode. Valid values are `cloud` or `onprem`. The default value is `onprem`.**
`oauth_audience`	**The audience claim of the access token.**
`oauth_issuer_whitelist`	**The list of valid JSON Web Token (JWT) issuers from whom the tokens are expected.**
`oauth_jwks_endpoint`	**The JWKS endpoint.**

Copyright © 2025 Ping Identity Corporation

A sample `abs.properties` file is displayed below.

Copyright © 2025 Ping Identity Corporation

```
# Ping Identity Corporation, ABS config file
# All the keys should be present, leave blank value if not applicable

# Set the timezone to utc or local. The default timezone is utc.
timezone=utc
# REST API port
management_port=8080
# Log levels (ALL > DEBUG > INFO > WARN > ERROR > FATAL > OFF)
log_level=DEBUG
# Java KeyStore password
jks_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCulhAUKj7ww4=
# MongoDB replica set URI. For example, mongodb://<IP1>:<Port>,<IP2>:<Port>,<IP3>:<Port>. Maximum three
nodes can be configured.
mongo_rs=mongodb://localhost:27017
# MongoDB Database
metadata_dbname=abs_metadata
data_dbname=abs_data
mldata_dbname=abs_mldata
# MongoDB authentication
# If authentication is not enabled in MongoDB, set the mongo_auth_mechanism to NONE
# The supported MongoDB authentication mechanisms are DEFAULT and PLAIN.
# If authentication mechanism is DEFAULT, provide MongoDB username and password for mongo_username
# and mongo_password. If authentication mechanism is PLAIN, provide external
# LDAP username and password in mongo_username and mongo_password.
mongo_auth_mechanism=DEFAULT
mongo_username=absuser
mongo_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCulhAUKj7ww4=
# Mongo DB SSL
# Set to true if Mongo DB instance is configured in SSL mode.
# By default, ABS will try to connect to Mongo using non-SSL connection
mongo_ssl=false
# Mongo DB Server Certificate Verification
# Set to true if Mongo DB instance is configured in SSL mode and you want to do the server certificate
verification
# By default ABS will not verify the MongoDB server certificate
mongo_certificate=false
# Job queue size per node
queue_size=10
# Setting as true makes an ABS node for dashboard query only and does not participate in ABS cluster for
log processing
dashboard_node=false
# Memory for webserver and streaming server (unit is in MB)
system_memory=4096
# E-mail alerts
enable_emails=false
# SMTP host
smtp_host=smtp.example.com
# SMTP port
smtp_port=587
# Set this value to true if smtp host support SSL
smtp_ssl=true
# Set this value to true if SSL certificate verification is required
smtp_cert_verification=false
# Sender email id
```

Copyright © 2025 Ping Identity Corporation

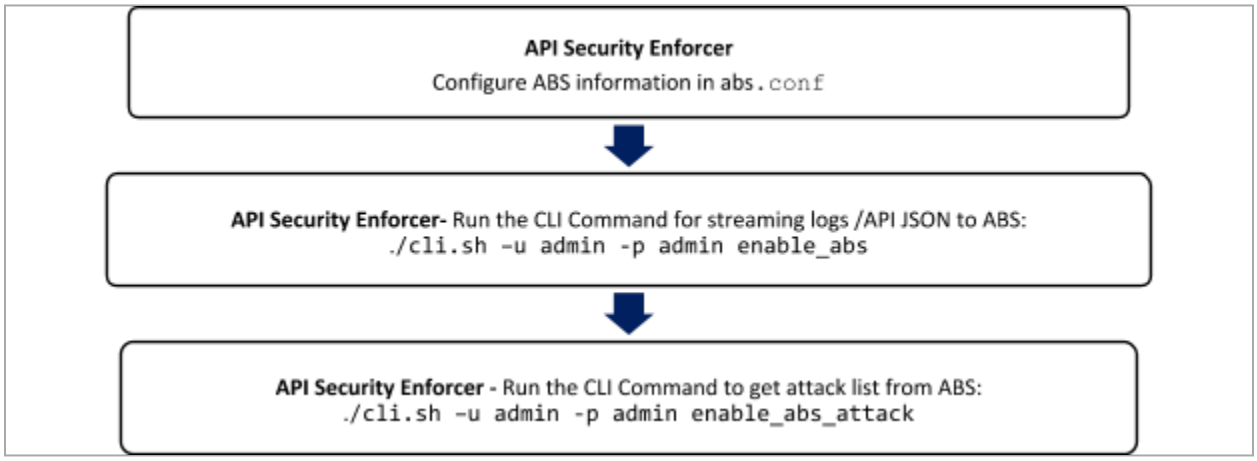
```
sender_email=sender@example.com
# Sender's email password
email_password=OBF:AES:UXzB+y+69Bn3xiX6N822ad4hf5IfNfJY9w==:T+QzM6qtc0+6MVsx4gU5p0LMHAI/y+w8DDsWv6VxVAk=
# Receiver's email id
receiver_email=receiver@example.com
# Set this value to appropriate AWS S3 Bucket name, if ABS is running in cloud mode
bucket_name=
# Set this value to appropriate Env Id, if ABS is running in cloud mode
env_id=
# Set this value to either cloud / onprem, as per the ABS running mode. By default set to onprem
deployment_type=onprem
# Token validation params
# Audience
oauth_audience=
# Issuer whitelist
oauth_issuer_whitelist=
# JWKS endpoint
oauth_jwks_endpoint=
```

Connect ABS to ASE

Before connecting API Behavioral Security (ABS), API Security Enforcer (ASE) must be installed.

For more information on installing and configuring ASE, see the ASE Admin guide.

The following diagram summarizes the process of connecting ABS to ASE:



The following is a sample `abs.conf` file which is part of the ASE:

Copyright © 2025 Ping Identity Corporation

```
; API Security Enforcer ABS configuration.
; This file is in the standard .ini format. The comments start with a semicolon (;).
; Following configurations are applicable only if ABS is enabled with true.
; a comma-separated list of abs nodes having hostname:port or ipv4:port as an address.
abs_endpoint=127.0.0.1:8080
; access key for abs node
access_key=OBF:AES://ENOzsqOEhDBWLDY+pIoQ:jN6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0
; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw:zzi77GIFF5xkQJccjIrIVWU+RY5CxUhp3NLcNBel+3Q
; Setting this value to true will enable encrypted communication with ABS.
enable_ssl=true
; Configure the location of ABS's trusted CA certificates. If empty, ABS's certificate
; will not be verified
abs_ca_cert_path=
```

The `access_key` and `secret_key` are the keys that were defined in the `abs_init.js` file when configuring MongoDB.

> **Note**
>
> To connect an ASE to ABS, configure the `abs.conf` file on any ASE in the cluster and run the **`CLI`** commands. This ensures all the ASE nodes in the cluster will be updated to connect with ABS.

If ABS is running in cluster mode, choose the Internet Protocol (IP) address and port from any ABS node to add to the `abs.conf` file in ASE.

Dataflow

ASE to the ABS node defined in `abs.conf` to obtain available ABS IP addresses (step 1). In stand-alone mode, ABS sends the only IP address. In cluster mode, ABS sends the IP addresses of all available ABS nodes to ASE.

After ASE receives the IP address, it establishes a session with ABS by sending the secret and access keys (step 2). After successful authentication, ASE streams the access log files and API JSON files to the ABS node (step 3). After sending the files, it receives the attack lists (only available if blocking is activated for ASE) from ABS (step 4). When the transaction is complete, ASE logs out from ABS (step 5).

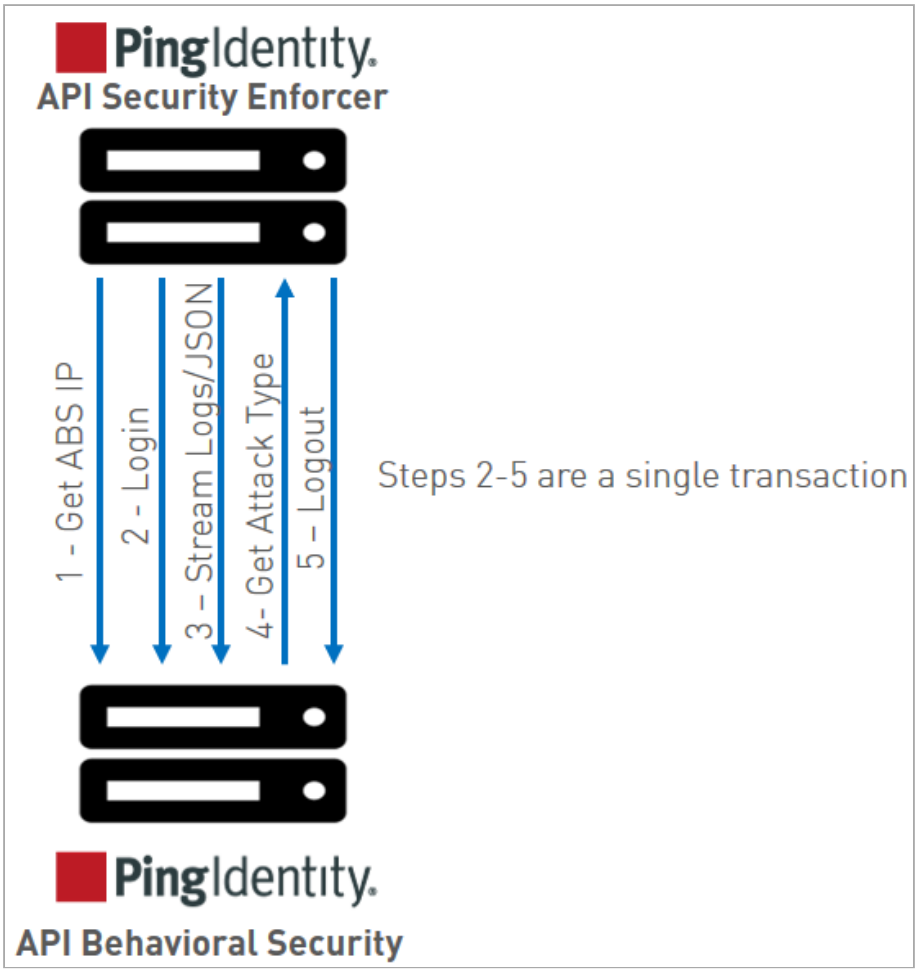
ABS uses machine learning algorithms to discover attacks, anomalies, and other traffic information. It stores incoming ASE logs and then passes these logs to the machine learning engine for analysis. In high load environments, a single ABS node may not be able to process all log files, and multiple ABS nodes should be deployed for log processing.

The following diagrams show the ASE – ABS Dataflow.

Stand-alone mode

In stand-alone mode, a single MongoDB node is used for both read and write operations. A stand-alone mode of deployment is only recommended for testing purposes.

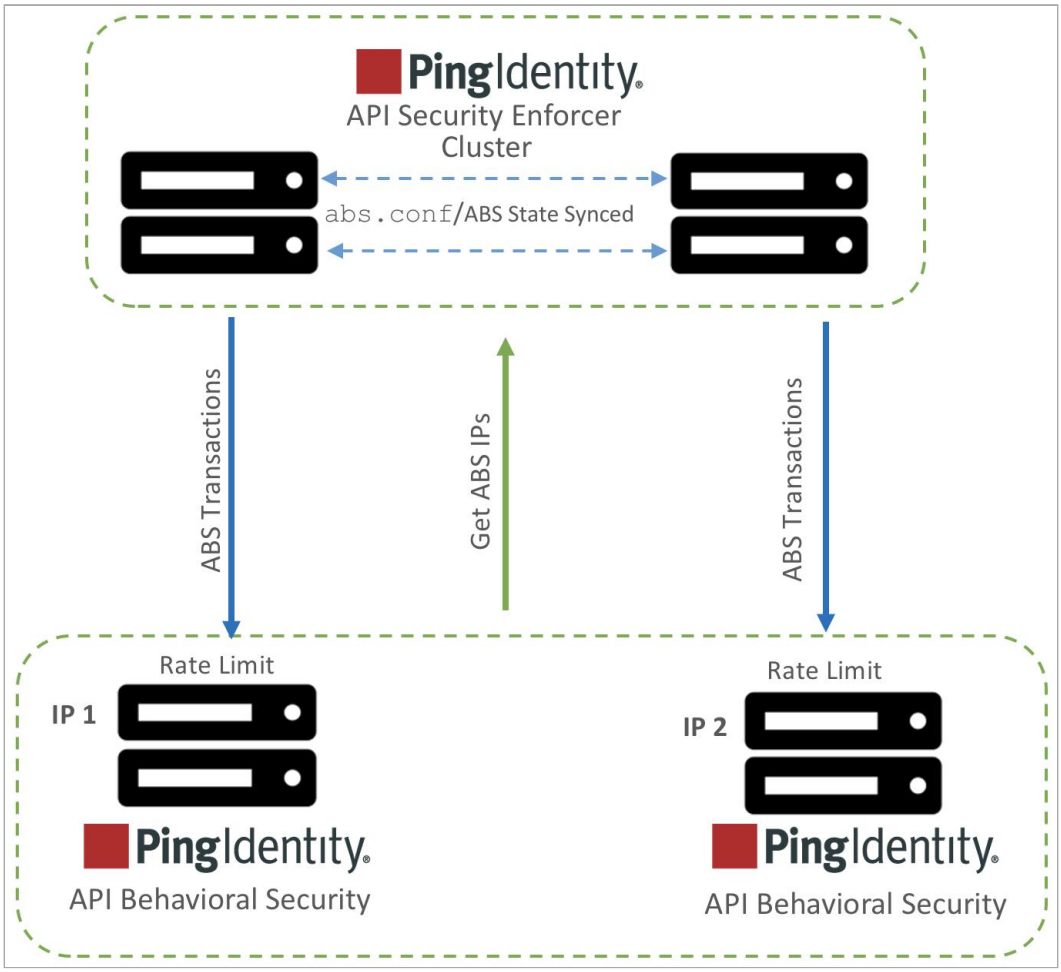
Copyright © 2025 Ping Identity Corporation



Cluster mode

In cluster mode, ASE nodes synchronize the `abs.conf` file as well as the state of each ABS node. The ABS cluster nodes do not communicate among themselves. Each node records its status in MongoDB and reads about the state of other nodes from the database.

Copyright © 2025 Ping Identity Corporation



ABS cluster

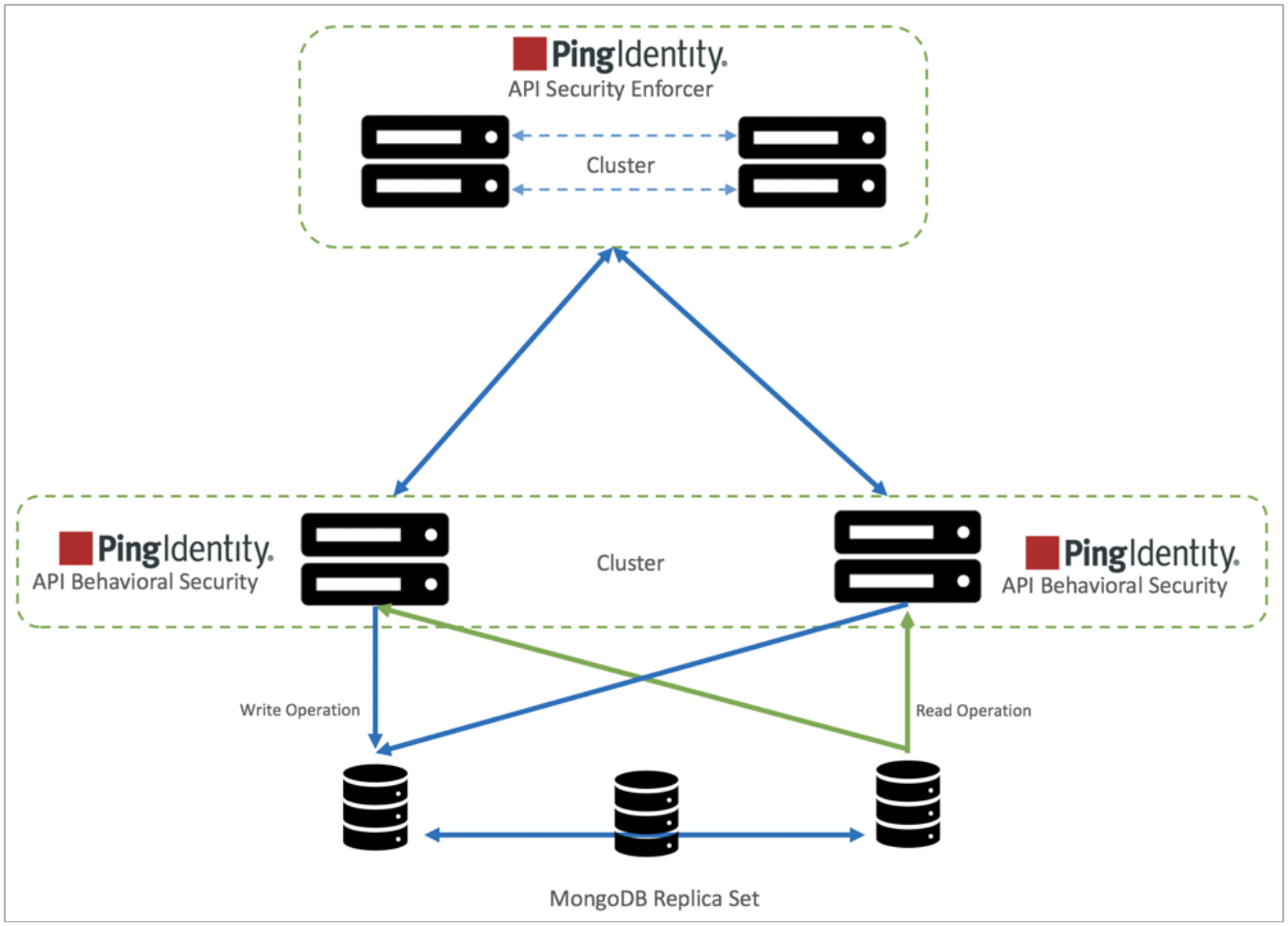
An API Behavioral Security (ABS) cluster consists of stateless ABS nodes communicating with a MongoDB replica set.

Each ABS node connects to the MongoDB cluster to obtain cluster configuration information that describes peer nodes. ABS nodes themselves do not communicate with each other; they periodically send heartbeats to MongoDB with status information. Each ABS node exposes:

- REST APIs for log streaming between ABS and API Security Enforcer (ASE)
- REST APIs between ABS and management applications which fetch metrics, anomalies, attack types, backend error, blocked connections, flow control, and cluster status.

An ABS cluster is depicted in the following diagram.

Copyright © 2025 Ping Identity Corporation



Configuring an ABS cluster

About this task

To configure an API Behavioral Security (ABS) cluster:

Steps

1. Install MongoDB in a replica set
2. Connect ABS to MongoDB
3. To create an ABS cluster, add an ABS node and connect it to MongoDB primary node.

> **Note**
>
> Since ABS forms a stateless cluster, the information of all the nodes in the cluster is fetched by ABS nodes from MongoDB.

4. To scale down ABS cluster, stop the ABS node that you wish to remove from the cluster.
    1. Edit the `abs.properties` file to remove MongoDB Internet Protocol (IP) address.

Copyright © 2025 Ping Identity Corporation

ABS logs

The active API Behavioral Security (ABS) log file `abs.log` is located in the logs directory and rotated every 24 hours at midnight local time.

The rotated log files append timestamps to the name and follow the naming convention of `abs.log.` *<yyyy>-<mm>-<dd>* (for example, `abs.log.2018-11-24`).

Here is an example:

```
-rw-r--r--. 1 root root 68K Apr 25 23:59 abs.log.2019-04-25
-rw-r--r--. 1 root root 68K Apr 25 23:59 abs.log.2019-04-24
-rw-r--r--. 1 root root 68K Apr 26 23:59 abs.log.2018-04-26
-rw-r--r--. 1 root root 158K Apr 27 23:59 abs.log.2018-04-27
-rw-r--r--. 1 root root 32K Apr 28 11:21 abs.log
```

The ABS log file contains `INFO` messages (for example, ABS started, MongoDB status) and `ERROR` messages (for example, MongoDB is not reachable). The log files also contains entry of all the email alerts sent.

Here is a snippet of an `abs.log` file:

```
2019-04-28 11:16:45 INFO - starting abs periodic actions
2019-04-28 11:16:45 INFO - MongoDB heartbeat success
2019-04-28 11:16:45 INFO - notification node not set.
2019-04-28 11:16:45 INFO - training period 1 hours.
2019-04-28 11:16:45 INFO - system threshold update interval 1 hour(s).
2019-04-28 11:16:45 INFO - api discovery interval 1 hour(s).
2019-04-28 11:16:45 INFO - subpath limit: 100
2019-04-28 11:16:45 INFO - ABS started successfully...
2019-04-28 11:17:45 INFO - MongoDB heartbeat success
2019-04-28 11:19:45 ERROR - MongoDB heartbeat failure
```

Purge the processed access logs from ABS

A `purge.sh` script either archives or purges processed access log files.

The access log files are stored in the `/opt/pingidentity/abs/data` directory.

> **Note**
>
> When the **`purge`** script is run, the processed access log files are permanently deleted from the `/opt/pingidentity/abs/data` directory. Always backup the files before deleting.

Located in the `/opt/pingidentity/abs/util` directory, the `purge` script deletes logs older than the specified number of days. Run the script using the API Behavioral Security (ABS) command line. For example:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/abs/util/purge.sh -d 3
In the above example, purge.sh deletes all access log files older than 3 days. Here is sample output.
/opt/pingidentity/abs/util/purge.sh -d 3
This will delete the data in /opt/ pingidentity/abs/data which is older than 3 days.
Are you sure (yes/no): yes
removing /opt/pingidentity/abs/data/2018-04-10-11_21/9k2unv5l2bsgurneot3s3pmt03/ : last changed at Mon Jan
10 11:32:31 IST 2018
removing /opt/ pingidentity/abs/data/2018-04-10-11_21/ilq67a3g5sve2pmpkkp271o37c/ : last changed at Mon Jan
10 11:32:31 IST 2018
```

External log archival

The `purge` script can also archive logs older than the specified number of days to secondary storage. Use the `-l` option and include the path of the secondary storage to archive log files. For example:

```
/opt/pingidentity/abs/util/purge.sh -d 3 -l /tmp/
```

In the above example, log files older than `3-days` are archived to the `tmp` directory. To automate log archival, add the script to a `cron` job.

Purge MongoDB data

The API Behavioral Security (ABS) MongoDB purge script dumps and/or deletes processed AI Engine and machine learning data from MongoDB.

It is recommended to archive the data before purging it. The `purge_mongo.sh` script is available in the `/<pi-install-dir>/pingidentity/abs/util` directory.

The script offers three options:

- Only purge data
- Only dump data
- Dump data into a specified directory and then purge it

Prerequisites-Ensure that the following prerequisites are fulfilled, before using the script:

- Execute the script from an ABS AI Engine node with connectivity to the MongoDB primary node.
- The necessary write permissions are available on the directory where the data dump is stored.
- Database names used as command line arguments like the data_dbname and mldata_dbname should be same as configured in ABS configuration - abs.properties file.

> **Note**
>
> It is recommended to execute the script during low load periods or during ABS down times.

The `purge_mongo.sh` script supports the following arguments:

Copyright © 2025 Ping Identity Corporation

Argument	Description
`--data_db <abs database>`	Use this argument to specify the name of the ABS database.
`--mldata_db <ml database>`	Use this argument to specify the name of the ML database.

> **Note**
> You must specify at least one database while executing the script. You can also specify both the databases in a single command.

`-d <days>` or `--days <days>`	Number of days of data to be retained. The default number of days is seven. The minimum number of days that can be specified is one and the maximum is 365.
`-l <path_to_dump_dir>` or `--location <path_to_dump_dir>`	The directory path to store the MongoDB dump.
`--purge_only`	Use this option when you only need to delete the data and not take a data dump.
`--dump_only`	Use this option when you need to take the data dump without deleting the data.
`--h` or `--help`	Use this argument for more information on the purge script parameters.
`--gzip`	Use this argument to compress the data dump. It can be used with `dump only` and `purge and dump` options. This argument cannot be used with `purge only` option.

The following sections show the sample usage of `purge_mongo.sh` script with the three options:

Purge data

The following are a few sample commands to purge the MongoDB data:

- ./purge_mongo.sh --data_db <*abs database*> -d <*days*> --purge_only
- ./purge_mongo.sh --mldata_db <*ml database*> -d <*days*> --purge_only
- ./purge_mongo.sh --data_db <*abs_database*> --mldata_db <*ml database*>d <*days*> --purge_only

For example, the following command deletes ABS data older than 80 days.

Copyright © 2025 Ping Identity Corporation

```
./purge_mongo.sh --data_db abs_data -d 80 --purge_only
Starting the purge mongo tool
This will delete the documents in  abs_data database that are older than 80 days.
Are you sure (yes/no): yes
Deleting the documents in  abs_data database that are older than 80 days.
Please see /opt/pingidentity/abs/logs/purge/purge.log.2020-11-16-05-01-42 for more details
```

Dump data

The following are a few sample commands to purge the MongoDB data:

- ./purge_mongo.sh --data_db <*abs database*> -d <*days*> -l <*path*> --dump_only
- ./purge_mongo.sh --mldata_db <*ml database*> -d <*days*> -l <*path*> --dump_only
- ./purge_mongo.sh --data_db <*abs_database*> --mldata_db <*ml database*>d <*days*> -l <*path*> --dump_only

For example, the following command dumps data older than 80 days from ml_database into a `/tmp` directory. It does not delete the data. `/tmp` is used as an example reference here, you can substitute `/tmp` with any other directory path in your environment.

```
./purge_mongo.sh --mldata_db abs_mldata -d 80 -l /tmp --dump_only
Starting the purge mongo tool
Storing abs ml data from mongo at /tmp/ml_mongo_data.2020-11-16-06-09-06
Please see /opt/pingidentity/abs/logs/purge/purge.log.2020-11-16-06-10-42 for more details
```

Purge and dump data

The following are a few sample commands to purge and dump the data:

- ./purge_mongo.sh --data_db <*abs database*> -d <*days*> -l <*path*>
- ./purge_mongo.sh --mldata_db <*ml database*> -d <*days*> -l <*path*>
- ./purge_mongo.sh --data_db <*abs_database*> --mldata_db <*ml database*> -d <*days*> -l <*path*>

For example, the following command dumps data older than 80 days from ml_database into a `/tmp` directory and deletes the data. `/tmp` is used as an example reference here, you can substitute `/tmp` with any other directory path in your environment.

```
./purge_mongo.sh --mldata_db abs_mldata -d 80 -l /tmp
Starting the purge mongo tool
Storing abs ml data from mongo at /tmp/ml_mongo_data.2020-11-16-06-12-14
This will delete the documents in  abs_mldata database that are older than 80 days.
Are you sure (yes/no): yes
Deleting the documents in  abs_mldata database that are older than 80 days.
Please see /opt/pingidentity/abs/logs/purge/purge.log.2020-11-16-06-12-42 for more details
```

> **Note**
>
> By default, the script dumps all data and then removes processed data older than seven days.

Copyright © 2025 Ping Identity Corporation

In case there is a failure in execution of `purge_mongo.sh` the script exits. You can retry executing the script. The execution details are logged in `/<pi-install path>/pingidentity/abs/logs/purge/` directory.

Resetting MongoDB

API Behavioral Security (ABS) AI engine provides a script to factory reset MongoDB data.

Before you begin

Make sure to take a backup of your current data before running the reset script. Once you run the MongoDB reset script, the deleted data cannot be retrieved.

The reset MongoDB script deletes all the documents from all the collections of `abs_data` and `abs_mldata` from MongoDB. The `reset_mongo.sh` script is available in the `/opt/pingidentity/abs/util` directory. Copy the script from the `util` directory to your MongoDB primary node.

About this task

To execute the script, you need the following information:

- MongoDB credentials: `mongo_username` and `mongo_password` configured in `abs.properties`.
- Database name and port number: `data_dbname`, `mldata_dbname`, and `mongo_master_port` configured in `abs.properties`.
- If your MongoDB installation is configured to use Secure Sockets Layer (SSL), use the --ssl option. The following examples assume that MongoDB is configured to use TLS.

Steps

1. For more information on the reset script parameters, run the reset help script from the MongoDB command line:

```
/opt/pingidentity/mongo/reset_mongo.sh –help
```

1. Reset ABS and machine learning data.

   Example:

   The following example resets both ABS and machine learning (ml) data:

```
/opt/pingidentity/mongo/reset_mongo.sh -u absuser -p abs123 --tls --data_db abs_data --
mldata_db abs_mldata --auth_db admin --port 27017
```

2. Reset only machine learning (ml) data.

   Example:

   The following example resets only the machine learning data:

```
/opt/pingidentity/mongo/reset_mongo.sh -u absuser -p abs123 --tls --mldata_db abs_mldata --
auth_db admin --port 27017
```

Copyright © 2025 Ping Identity Corporation

3. Reset only ABS data.

   Example:

   The following example resets only the ABS data:

   ```
   /opt/pingidentity/mongo/reset_mongo.sh -u absuser -p abs123 --tls --data_db abs_data --auth_db
   admin --port 27017
   ```

2. Run the MongoDB script.

   Result:

   The following snippet shows the output when the reset MongoDB script is run:

Copyright © 2025 Ping Identity Corporation

```
./reset_mongo.sh -u absuser -p abs123 --port 27017 --data_db abs_data --mldata_db abs_mldata --tls
Please make sure that there is no ABS process running before running the reset_mongo script.
Are you sure you want to continue... (yes/no): yes
This will delete all the documents in  abs_data database
Are you sure? (yes/no): yes
Deleting the documents in  abs_data database.
2019-10-11T05:46:43.726+0000 W  CONTROL  [main] Option: ssl is deprecated. Please use tls instead.
2019-10-11T05:46:43.727+0000 W  CONTROL  [main] Option: sslAllowInvalidCertificates is deprecated.
Please use tlsAllowInvalidCertificates instead.
MongoDB shell version v4.2.0
connecting to: mongodb://127.0.0.1:27017/?
authSource=admin&compressors=disabled&gssapiServiceName=mongodb
2019-10-11T05:46:43.802+0000 W  NETWORK  [js] TLS peer certificate validation failed: self signed
certificate
Implicit session: session { "id" : UUID("400fcaa5-57dd-4123-a5e6-b54c1e0bdfda") }
MongoDB server version: 4.2.0
switched to db abs_data

Removing all documents of all collections in ABS_DATA
Removing all documents from  [abs_data.api_attack_dos_anomaly]
Removing all documents from  [abs_data.api_config.chunks]
Removing all documents from  [abs_data.api_config.files]
Removing all documents from  [abs_data.api_json]
Removing all documents from  [abs_data.api_key_metrics]
Removing all documents from  [abs_data.attack_management]
Removing all documents from  [abs_data.attack_management_audit]
Resetting the [abs_data.attack_ttl] to default values
Removing all documents from  [abs_data.backend_errors]
Removing all documents from  [abs_data.bc_summary]
Removing all documents from  [abs_data.blocked_connections]
Removing all documents from  [abs_data.discovered_apis]
Removing all documents from  [abs_data.discovery_api_metadata]
Removing all documents from  [abs_data.discovery_ir.chunks]
Removing all documents from  [abs_data.discovery_ir.files]
Removing all documents from  [abs_data.extended_ml_threshold]
Removing all documents from  [abs_data.extended_trained_model]
Removing all documents from  [abs_data.extended_training_model]
Removing all documents from  [abs_data.external_ioc_type]
Removing all documents from  [abs_data.internal_ioc]
Removing all documents from  [abs_data.internal_ioc_audit]
Removing all documents from  [abs_data.ioc]
Removing all documents from  [abs_data.ioc_anomaly]
Removing all documents from  [abs_data.ir.chunks]
Removing all documents from  [abs_data.ir.files]
Removing all documents from  [abs_data.log_nodes]
Removing all documents from  [abs_data.ml_result]
Removing all documents from  [abs_data.ml_threshold]
Removing all documents from  [abs_data.notifications]
Removing all documents from  [abs_data.oauth_metrics]
```

Copyright © 2025 Ping Identity Corporation

> **Note**
>
> The reset script does not delete the following meta data:
> - ABS cluster information
> - ABS configuration
> - Global configuration from the `abs_init.js` file
> - Scale configuration from the `abs_init.js` file
> - Dictionary generated by ABS AI engine

3. **To verify that the MongoDB reset script executed successfully, run the ABS Admin REST API.**

   The output should not show any API Security Enforcer ASE access log and API information. It should only display ABS cluster information, MongoDB primary and secondary, and client identifier TTL value reset to zero.

   Result:

   The following is a sample output of Admin API after MongoDB reset script is run:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, and ASE logs",
    "across_api_prediction_mode": false,
    "api_discovery": {
        "subpath_length": "1",
        "status": true
    },
    "abs_cluster": {
        "abs_nodes": [
            {
                "node_ip": "172.16.40.19",
                "os": "Red Hat Enterprise Linux Server",
                "cpu": "16",
                "memory": "62G",
                "filesystem": "1%",
                "bootup_date": "Thu Oct 10 10:08:37 UTC 2019"
            }
        ],
        "mongodb_nodes": [
            {
                "node_ip": "172.16.40.236:27017",
                "status": "secondary"
            },
            {
                "node_ip": "172.16.40.237:27017",
                "status": "secondary"
            },
            {
                "node_ip": "172.16.40.235:27017",
                "status": "primary"
            }
        ]
    },
    "percentage_diskusage_limit": "80%",
    "scale_config": {
        "scale_up": {
            "cpu_threshold": "70%",
            "cpu_monitor_interval": "30 minutes",
            "memory_threshold": "70%",
            "memory_monitor_interval": "30 minutes",
            "disk_threshold": "70%",
            "disk_monitor_interval": "30 minutes"
        },
        "scale_down": {
            "cpu_threshold": "10%",
            "cpu_monitor_interval": "300 minutes",
            "memory_threshold": "10%",
            "memory_monitor_interval": "300 minutes",
            "disk_threshold": "10%",
            "disk_monitor_interval": "300 minutes"
        }
    },
    "attack_ttl": {
        "ids": [
            {
                "id": "ip",
                "ttl": 0
            },
```


Copyright © 2025 Ping Identity Corporation

```
            {
                "id": "cookie",
                "ttl": 0
            },
            {
                "id": "access_token",
                "ttl": 0
            },
            {
                "id": "api_key",
                "ttl": 0
            },
            {
                "id": "username",
                "ttl": 0
            }
        ]
    }
}
```

Adding a member to an existing MongoDB replica set

You can add a new node to an existing MongoDB replica set.

Before you begin

You must have:

- An active replica set.
- A new MongoDB system accessible by the replica set.
- ClusterAdmin privileges.

About this task

> **Note**
>
> `absrs01` is the name of the replica set used in the following steps.

To add a node to an existing replica set:

Steps

1. Create the MongoDB directory structure on the new MongoDB node:

```
# mkdir -p /opt/pingidentity/mongo/data/opt/pingidentity/mongo/logs \ /opt/pingidentity/mongo/key
```

2. Download MongoDB 4.2 on the node and extract it to `/opt/pingidentity/mongo`:

```
# cd /opt/pingidentity/ /opt/pingidentity# wget \ https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.2.0.tgz \ -O mongodb.tgz && tar xzf mongodb.tgz -C /opt/pingidentity/mongo/ --strip-components=1
```


Copyright © 2025 Ping Identity Corporation

3. Update the shell path variable and reload the shell:

```
/opt/pingidentity# echo PATH=$PATH:/opt/pingidentity/mongo/bin >> ~/.bashrc; /opt/pingidentity#
source ~/.bashrc
```

4. Copy the contents of the `/opt/pingidentity/mongo/key` directory from the primary node to the new node into `/opt/pingidentity/mongo/key`.

5. Start the MongoDB database on the new node:

```
/opt/pingidentity# cd mongo /opt/pingidentity/mongo# mongod --auth --dbpath ./data/ --logpath ./
logs/mongo.log --port 27017 --replSet absrs01 --fork --keyFile ./key/mongodb-keyfile -bind_ip
0.0.0.0
```

6. Connect to the `mongo` shell of the primary node and run the following command:

```
absrs01:PRIMARY> rs.add({"host": "<IP address of new node>:27017", "priority": 2})
```

Result:

The state of the new node changes to `STARTUP2`. This indicates that the synchronization between the replica set and the new node has started.

7. Verify that the new node is added as a secondary node to the replica set using the following command:

```
absrs01:PRIMARY> rs.status()
```

Next steps

For more information, see https://docs.mongodb.com/manual/tutorial/expand-replica-set/.

Removing a member from MongoDB replica sets

You can remove a node from an existing MongoDB replica set.

Before you begin

You must have:

- An active replica set.
- ClusterAdmin privileges.

About this task

To remove a node from an existing replica set:

Steps

1. Connect to the node that you want to remove and shut down the MongoDB on it using the following command:

Copyright © 2025 Ping Identity Corporation

```
absrs01:PRIMARY> db.shutdownServer()
```

2. Connect to the primary member of the replica set and run the following command to remove the node:

```
absrs01:PRIMARY> rs.remove("<IP Address or hostname of the node to be removed>:27017")
```

Next steps

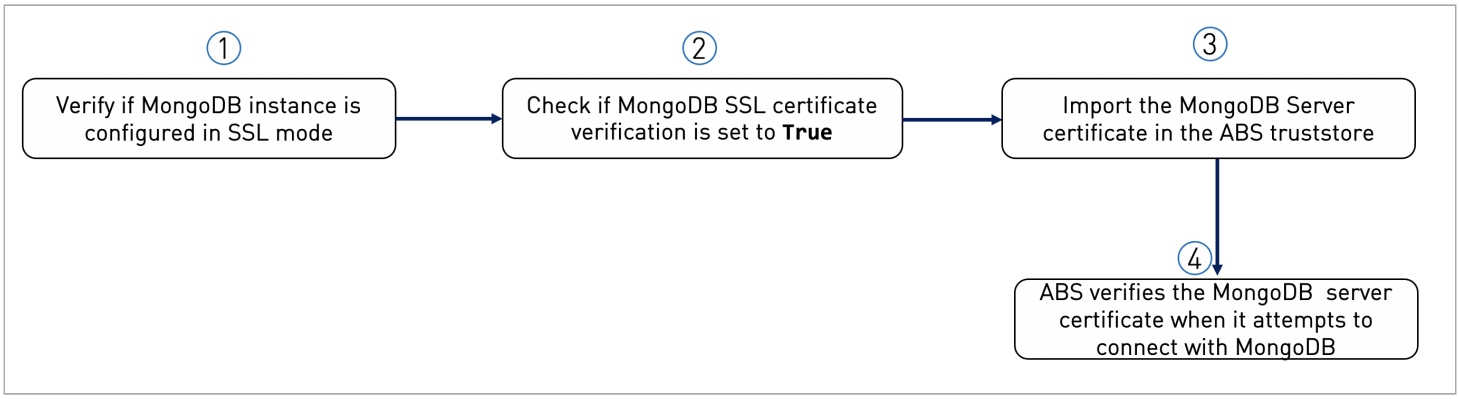
For more information, see Remove Members from Replica Set in the MongoDB documentation.

Verify MongoDB SSL certificates

You can configure API Behavioral Security (ABS) to verify the validity of MongoDB server certificate, when it tries to connect with MongoDB.

About this task

This is an optional check. and the following diagram shows the summary of steps involved in this verification.



Note

Ensure the following steps are completed, so that ABS can verify MongoDB server certificate before connecting to it.

Steps

1. To verify the validity of MongoDB, check if the `mongo_ssl` parameter in the `/<pi_install_path>/pingidentity/abs/config/abs.properties` file is set true.
2. Check if the `mongo_certificate` parameter in the `/<pi_install_path>/pingidentity/abs/config/abs.properties` file is set true.
3. Import the MongoDB Server certificate into the `abs.jks` truststore, using either of the following commands as applicable. The commands prompt for a `destination keystore password`, and the password entered should be same as the `jks_password` configured in the abs.properties file.

   Choose from:

Copyright © 2025 Ping Identity Corporation

```
# keytool -import -file  <mongodb-cert.crt>  -storetype JKS -keystore /<pi_install_path>/
pingidentity/abs/config/ssl/abs.jks
```

If the MongoDB server certificate is in `.pem` format, use the following command to import the certificate in to the ABS truststore.

```
# keytool -import -v -trustcacerts -file server.pem -keystore /<pi_install_path>/pingidentity/abs/
config/ssl/abs.jks -storetype JKS
```

When ABS starts, it loads the certificates available in `abs.jks` truststore. If the server certificate presented by MongoDB gets validated, ABS connects with it and completes the booting.

If the SSL server certificate verification fails, ABS will not start and a `CertificateException` is thrown by ABS. The error is logged in `/<pi_install_ path>/pingidentity/abs/abs.log.`

> **Note**
>
> If ABS is running and the MongoDB server certificate expires in between, it will not stop. An error message is logged in `/<pi_install_ path>/pingidentity/abs/abs.log.`
> You can also use a CA-signed certificate to verify the MongoDB server certificate. For that, import your existing CA-signed certificate into ABS by following the instructions explained in Import existing CA-signed certificates. Once the certificate is imported, complete Step 1 through Step 3 above so that ABS can verify MongoDB server certificate.

Email alerts and reports

API Behavioral Security (ABS) sends e-mail notifications.

E-mail notifications are classified into two categories:

- Alerts – event-based updates to notify administrators of potential issues
- Reports – standard reports sent every 24 hours at midnight

Email parameters in `abs.properties` correspond to your e-mail server. By default, e-mail notifications are disabled. Enable notifications after configuring e-mail IDs and server.

> **Note**
>
> If you want more than one person to be notified, use an email alias in **sender_email** field.

Copyright © 2025 Ping Identity Corporation

```
#Enable or Disable e-mail alerts
enable_emails=false
#Provide the details of sender and receiver of e-mail
#Sender's e-mail ID
sender_email=mail@yourdomain.com
#Sender's e-mail password
email_password=mypassword
#Receiver's e-mail ID
receiver_email=mail@yourdomain.com
#SMTP port
smtp_port=587
#SMTP host
smtp_host=smtp.smtphost.com
```

ABS alerts

API Behavioral Security (ABS) email alerts are sent based on the following category of events.

These events are also logged in the `abs.log` file. The threshold values for these events are pre-set. If you want to change the threshold values after the system is running, then you have to manually change the values in MongoDB.

- Dynamic Rate Limit: alert sent when CPU, disk, or memory crosses the configured threshold value.
- ABS Node: alert sent when ABS cluster nodes are added or removed.
- MongoDB: alert sent when a MongoDB node is added or becomes inaccessible.
- Percentage Disk Usage Limit: alert sent when the disk usage reaches the configured `percentage_diskusage_limit` value. When this limit is reached, ABS stops accepting any new access log files from ASE. The alert is also logged in the `abs.log` file. You can use `update.sh` script in `/abs/util/` directory to update the thresholds for `Percentage Disk Usage Limit`.
- License: The following license related alerts are sent:
    - ABS license invalid: alert is sent if the ABS license is found to be invalid. In this case ABS shuts down.
    - ABS license expiration: alert sent when ABS license is expired.
    - Transaction limit reached: alert sent when ABS reaches the licensed monthly transaction limit.
- Scale Up and Scale Down: alert sent when a system resource, such as CPU, memory, or disk utilization, is above or below its threshold value for a specified interval of time. If the value is above the threshold value, add ABS nodes to distribute the load. If the resource utilization is below the lower threshold, you may remove an ABS node from the ABS cluster.
- DDoS attack alert: ABS sends alerts for multi-client Login Attacks and for API DDoS Attack Type 1. The email alert provides a time period for the attack along with a URL to access information on all client IPs participating in the attack.

Following is a template for alerts:

Copyright © 2025 Ping Identity Corporation

```
Event:  <the type of event>
Value:  <the specific trigger for the event>
When:  <the date and time of the event>
Where:  <the IP address of the server where the event occured>
```

For example,

```
Event: Scale Down ABS Node
Value : 192.168.11.166
CPU scale down threshold reached.
When : 2019-Jun-05 18:02:33 UTC
Where: 192.168.11.166
```

The following table describes the various email alerts sent by ABS and their possible resolution. The resolution provided is only a starting point to understand the cause of the alert. If ABS is reporting an alert even after the following the resolution provided, contact Ping Identity support.

Email alert	Possible cause and resolution
File System Maxed Out - Rate Limit Alert	Cause: A possible reason for this alert could be that historical access log files from ASE have accumulated on the storage disk. Resolution: Purge or archive the old access log files from storage disk.
ABS node added to cluster	ABS sends an email alert when a node joins an ABS cluster. Confirm: ABS admin should verify whether the correct ABS node joined the ABS cluster.
ABS node removed from cluster	ABS sends an email alert when a node is removed from an ABS cluster. Confirm: ABS admin should check the reason for removal of ABS node from the cluster. ABS node could disconnect from cluster because of network issues, a manual stop of ABS, or change in IP address of the ABS machine.
Memory scale up or scale down	Cause: ABS sends an email alert when the ABS node reaches the memory scale up or scale down limits in the configuration. The reason for reaching scale up limit can be because of large number of access log files coming from ASE. Scale down limit could be reached because of low number of access logs coming from ASE. Resolution: If ABS reaches scale up limit, add another ABS node to the cluster. If the system utilization is low, you can remove an ABS node from the cluster.
CPU scale up or scale down	Cause: ABS sends an email alert when the ABS node reaches the CPU scale up or scale down limits in the configuration. The reason for reaching scale up limit can be because of large number of access log files coming from ASE. Scale down limit could be reached because of low number of access logs coming from ASE. Resolution: If ABS reaches scale up limit, add another ABS node to the cluster. If the system utilization is low, you can remove an ABS node from the cluster.

Copyright © 2025 Ping Identity Corporation

Email alert	Possible cause and resolution
Disk scale up or scale down	Cause: ABS sends an email alert when the ABS node reaches the disk scale up or scale down limits in the configuration. The reason for reaching scale up limit can be because of large number of access log files coming from ASE. Scale down limit could be reached because of low number of access logs coming from ASE. Resolution: If ABS reaches scale up limit, add another ABS node to the cluster. If the system utilization is low, you can remove an ABS node from the cluster.
License <*path*> is invalid. ABS will shut down now	Cause: ABS sends this email alert when ABS does not have correct permissions to read the license file from the configured path, or there is a typing error in the name of the license file. Resolution: Validate current license file path. Also check for file permissions of the license file.
ABS license at <*path*> has expired. Please renew your license.	Cause: ABS sends this email alert when ABS license has expired. The license expires at the end of the license period. Resolution: Renew your ABS license.
Maximum transaction limit reached for the current month	ABS sends this warning message when ABS crosses the licensed monthly transaction limit.
API DDoS Attack Type 1 or Login DoS detected between <*timestamp*> and <*timestamp*> on node <*value*>	ABS sends this warning message when it detects an API DDoS attack type 1 or a Login DoS attack.
MongoDB primary node is down	Cause: ABS sends this email alert when MongoDB process is unavailable due to a shortage in memory or CPU. This alert can also trigger because of network issues for MongoDB node. Resolution: Check MongoDB Primary node status to bring it back online or add additional secondary node if needed.

ABS reports

API Behavioral Security (ABS) sends an e-mail report every 24 hours at midnight, 00:00:00 hours (local system time).

Each report includes values for the following parameters:

- ABS Node Status: resource utilization of CPU, file system, and operating system
- ASE Logs Processed: Compressed file size of ASE logs processed in 24 hours
- Total Requests: The number of requests in the processed log files in 24 hours
- Success: The total number of requests which got a 200-OK response
- Total Anomalies: Total number of anomalies detected across APIs in 24 hours
- Total IOC: Total number of attacks detected in 24 hours
- When: The time when the email report was sent

Copyright © 2025 Ping Identity Corporation

- Where: The ABS node that sent the email report
- MongoDB node IP address and status

Following is a sample ABS email template:

```
Dear DevOps,
    Please find the daily report generated by 192.168.11.166 at 2019-Jun-25 00:02:00 UTC
==================Cluster Details=============
ASE Logs Processed: 93.78MB
Total Request: 678590
Success: 596199
Total Anomalies: 7
Total IOC: 2
When : 2019-Jun-25 00:02:00 UTC
Where: 192.168.11.166

==================Node1 ===================
Host : 192.168.11.166
OS : Red Hat Enterprise Linux Server release 7.5 (Maipo)
CPU : 24
Memory : 62G
Filesystem : 39%
==========================================

================Mongo1 ====================
Host : 192.168.11.162
Status : up
==========================================

================Mongo2 ====================
Host :  192.168.11.164
Status : up
==========================================

================Mongo3 ====================
Host :  192.168.11.1685
Status : up
==========================================

==========================================
Best,
API Behavioral Security.
```

Global configuration update REST API

API Behavioral Security (ABS) provides a REST application programming interface (API) to update global configurations related to ABS AI engine.

The updated global configuration values take effect immediately. Following is the list of global configurations that you can update using the `globalconfig` API:

- `attack_initial_training`

Copyright © 2025 Ping Identity Corporation

- `attack_update_interval`
- `api_discovery`
- `discovery_initial_period`
- `discovery_subpath`
- `discovery_update_interval`
- `poc`

> **Note**
>
> The `poc` variable is used to set the ABS AI engine in POC (proof-of-concept) mode to demonstrate the capabilities of the AI engine. In a production environment the value is always set to `false` . It is recommended not to switch the mode in production environment. If you must change the ABS AI engine to `poc` mode, contact PingIndentity support.

- `url_limit`
- `response_size`
- `continous_learning`
- `attack_list_count`
- `root_api_attack`
- `session_inactivity_duration`

You can use the `globalconfig` API with GET and PUT methods. Following is the URL for `globalconfig` API. Only the Admin user can use the PUT method to update the values. For more information on different ABS users, see ABS users for API reports.

URL - https//*<abs_host>*:*<abs_port>*/v5/abs/globalconfig

	Header	Value
Access Key	`x-abs-ak`	***<string>*** **For example, `abs_ak` or the value of the access key that you configured at the time of installation.**
Secret Key	`x-abs-sk`	***<string>*** **For example, `abs_sk` or the value of the secret key that you configured at the time of installation.**

When you use the `globalconfig` API with GET method, it fetches the current value of the global configuration.

Copyright © 2025 Ping Identity Corporation

```
{
    “company”: “ping identity”,
    “name”: “api_globalconfig”,
    “description”: “This report contains status information of ABS global configurations”,
    “global_config”: {
        “attack_initial_training”: 2,
        “attack_update_interval”: 1,
        “api_discovery”: true,
        “discovery_initial_period”: 100,
        “discovery_subpath”: 3,
        “discovery_update_interval”: 1,
        “poc”: false,
        “url_limit”: 120,
        “response_size”: 150,
        “continuous_learning”: false,
        “attack_list_count”: 400000,
        “root_api_attack”: true,
        “session_inactivity_duration”: 10
    }
}
```

You can update the global configuration values that the API fetched using the PUT method. Provide the new values in the body as shown in the example below.

```
{
    "api_discovery": true,
    "discovery_initial_period": 1
    "discovery_update_interval": 1
 }
```

```
{
    "success": "global config updated successfully"
}
```

You can update either one or more than one global configurations at once. Note that the values are updated only when the body of the request is well-formed.

> **Note**
>
> You can also update the global configurations using Discovered APIs in PingIntelligence Dashboard or ABS Postman collections. However, you can only change the following variables using PingIntelligence Dashboard:

- `attack_initial_training`
- `attack_update_interval`
- `api_discovery`
- `discovery_initial_period`
- `discovery_subpath`
- `discovery_update_interval`

Copyright © 2025 Ping Identity Corporation

Related links

- Managing AI engine training
- AI engine training variables
- Update the training variables
- API discovery and configuration
- Managing discovery intervals

Indicators of Attacks on REST APIs

PingIntelligence detects and reports on Indicators of Attack (IoAs), which represent anomalous behavior on each API.

Detailed information about each IoA is provided in the description. Examples include:

- Data extraction or theft
- Anomalous API access patterns
- Credential stuffing and password spraying
- Account takeover with compromised credentials
- Broken object or function level authorization
- Data or command injection
- Abnormal API sequence
- Query string or header manipulation
- Probing and fuzzing attacks
- Extreme client activity

For each API, the API `JavaScript Object Notation (JSON)` file (see API Security Enforcer Admin Guide for information) determines whether the Indicators of Attack (IoA) and other reports are associated with an OAuth token, API Key, username, cookie, or Internet Protocol (IP) address. An environment with multiple APIs can support a mixture of identifier types in a single AI Engine.

Client identifier examples include:

- Tokens – When an API `JSON` file for an API is configured with OAuth2 token `parameter = true`, then the AI Engine builds models based on token activity on the API. After the API is trained, the AI Engine then detects and reports on IoAs associated with the tokens used by clients accessing the API. Analyzing token activity is recommended when access tokens are present as it is a unique client identifier that eliminates the issue of multiple clients sharing an IP address behind a gateway..

Copyright © 2025 Ping Identity Corporation

- API Keys – When the API `JSON` file for an API is configured with an API Key either in the query string or the header, then the AI Engine trains and detects IoAs based on the API Key values. For example, if there are two API Keys in the system, X-API-KEY-1 and X-API-Key-2 with values as `api_key_1` and `api_key_2`, then a total of four client identifiers are added to deny list of ASE:
    - X-API-KEY-1: `api_key_1`
    - X-API-KEY-2: `api_key_2`
    - X-API-KEY-1: `api_key_2`
    - X-API-KEY-2: `api_key_1`
- APIs with cookie – When the cookie parameter is configured, most IoAs are reported with cookie identifiers, the exception being pre-authentication attacks (such as client login attacks). Configuring the cookie parameter is recommended when cookies are present as it is a unique client identifier that eliminates issues identified below with IP addresses.
- Cookies – When the API `JSON` file for an API is configured with the cookie parameter, the the AI Engine builds models and detects IoAs for cookie identifiers, the exception being pre-authentication attacks (such as client login attacks). Configuring the cookie parameter is recommended when cookies are present as it is a unique client identifier that eliminates the issue of multiple clients sharing an IP address behind a gateway.
- IP Address –When neither the cookie nor token parameters are configured, all IoAs are reported with the client IP address which is determined based on the following:
    - XFF header present: The first IP address in the XFF list is used as the client identifier. When forwarding traffic, load balancers and other proxy devices with XFF enabled add IP addresses to the XFF header to provide application visibility of the client IP address. The first IP address in the list is typically associated with the originating IP address.

      > **Note**
      >
      > XFF is not always a reliable source of the client IP address and can be spoofed by a malicious proxy.

    - No XFF header: When no XFF header is present, the source IP address of the incoming traffic is used as the client identifier. In this configuration, make sure that the incoming traffic is using public or private IP addresses associated with the actual client devices, not a load balancer or proxy device on your premise.

      > **Note**
      >
      > When a load balancer or other proxy without XFF enabled is the source of the inbound traffic, then all client traffic will be associated with the load balancer IP addresses. This configuration will not provide effective attack reporting unless cookies or tokens are used.

- Usernames – Unlike other client identifiers, username is not configured in the API `JSON` file. Username information is captured in the following ways:
    - When the incoming request has a JSON Web Token (JWT), the user name can be extracted by ASE. For more information, see Extract user information from JWT in sideband mode or Extract user information from JWT in inline mode.

Copyright © 2025 Ping Identity Corporation

- When the incoming request has a custom header with user information, then the username is extracted from the custom header. For more information see, Extract username from custom header in sideband mode or Extract username from custom header in inline mode.
- When deployed in sideband mode, a platform (for example API Gateway) which supports capturing username information through token introspection can pass the user name through its sideband calls. API gateway integrations that support sending username information to PingIntelligence include:
  - Akana API gateway sideband integration
  - Axway sideband integration
  - Apigee integration
  - MuleSoft sideband integration
  - PingAccess sideband integration
  - NGINX sideband integration
  - NGINX Plus sideband integration
  - WSO2 integration

> **Important**
>
> The AI engine will not detect username attacks for requests where the server responds with an HTTP `401 Unauthorized Error` code. This will prevent blocking of a valid user if an attacker tries to impersonate the user. Also while reporting an abnormal sequence, if username is available with the API ecosystem, the AI engine reports username or else it reports OAuth token.

To change the client identifier used for IoA detection for an existing API, update the API JSON configuration to provide the desired client identifier. To re-train the model for this API, click on the Dashboard reset training in PingIntelligence Dashboard or save the API `JSON` file with a new name.

WebSocket API attack detection

In each application programming interface (API), the presence of the cookie parameter in the API JSON file (see API Security Enforcer Admin Guide for information) determines whether attacks are reported based on cookie identifier or Internet Protocol (IP) address.

> **Note**
>
> WebSocket API attack detection is only supported when ASE is running in Inline mode.

Copyright © 2025 Ping Identity Corporation

Client identifier determination – IP address or cookie

An environment with multiple APIs can support a mixture of identifier types in a single API Behavioral Security (ABS) system. Use cases include the following:

- API JSON with cookie parameter – When the cookie parameter is configured, most attacks are reported with cookie identifiers, the exception being pre-authentication attacks (for example, client login attacks). Configuring the Cookie parameter is recommended when cookies are present as it is a unique client identifier that eliminates the issues identified below with IP addresses.
- API JSON without cookie parameter – When the cookie parameter is not configured, all the attacks are reported with the client IP address which is determined based on the following:
- XFF header present: The first IP address in the XFF list is used as the client identifier. When forwarding traffic, load balancers and other proxy devices with XFF enabled add IP addresses to the XFF header to provide application visibility of the client IP address. The first IP address in the list is typically associated with the originating IP address.

> **Note**
>
> XFF is not always a reliable source of the client IP address and can be spoofed by a malicious proxy.

- No XFF header: When no XFF header is present, the source IP address of the incoming traffic is used as the client identifier. In this configuration, make sure that the incoming traffic is using public or private IP addresses associated with the actual client devices, not a load balancer or proxy device on your premise.

> **Note**
>
> When a load balancer or other proxy without XFF enabled is the source of the inbound traffic, then all client traffic will be associated with the load balancer IP addresses. This configuration will not provide effective attack reporting.

To change from a cookie to an IP identifier for an existing API, save the API JSON with a new name. ABS then re-trains the model for this API and starts detecting IP-based attacks. For more information on configuring API JSON files, see API Security Enforcer Admin Guide.

> **Note**
>
> OAuth2 token based attacks are not reported for WebSocket APIs.

The following tables list the attacks detected by ABS for WebSocket APIs for cookie and IP.

Cookie based detected attacks

Attack Type	Description	id
Summary Attack Report	Provides a summary of all attacks detected.	0
WS Cookie Attack	WebSocket session management service receiving an abnormal number of cookies.	50

Copyright © 2025 Ping Identity Corporation

Attack Type	Description	id
WS DoS Attack	Inbound streaming limits exceeded on a WebSocket service.	52
WS Data Exfiltration Attack	Data is being extracted via a WebSocket API service.	53

IP based detected attacks

Attack Type	Description	id
Summary Attack Report	Provides a summary of all attacks detected.	0
WS Identity Attack	WebSocket identity service receiving excessive upgrade requests.	51
WS DoS Attack	Inbound streaming limits exceeded on a WebSocket service.	52
WS Data Exfiltration Attack	Data is being extracted via a WebSocket API service.	53

Attack detection on root API

A root application programming interface (API) in API Security Enforcer (ASE) is defined by configuring `/` for <*url*> variable and `*` for <*hostname*>variable.

Following is a snippet of a truncated API JavaScript Object Notation (JSON) in ASE depicting the configuration of root API.

```
{
 "api_metadata": {
   "protocol": "http",
    "url": "/",
    "hostname": "*",
```

You can choose between enabling or disabling attack detection on global API by configuring <*root_api_attack*> global variable in the `abs_init.js` and `abs_init_ldap.js` file. By default attack detection is disabled on root API. Set it to `true` if you want to detect attacks on the root API. Configure this variable either before starting API Behavioral Security (ABS), or you can use the `update.sh` script to update the value. For more information on `update.sh` script, see Update the training variables

Copyright © 2025 Ping Identity Corporation

```
db.global_config.insert({
        "attack_initial_training": "24",
        "attack_update_interval": "24",
        "url_limit": "100",
        "response_size": "100",
        "job_frequency" : "10",
        "window_length" : "24",
        "enable_ssl": true,
        "api_discovery": false,
        "discovery_initial_period" : "24",
        "discovery_subpath": "1",
        "continuous_learning": true,
        "discovery_update_interval": "1",
        "attack_list_count": "500000",
        "resource_monitor_interval" : "10",
        "percentage_diskusage_limit" : "80",
         "root_api_attack" : false,
        "session_inactivity_duration" : "30"
});
```

Training and attack detection: If the attack detection is disabled on the root API, then ABS Admin REST API displays `n/a` (not applicable) for `training_started_at` and `training_duration`. The `prediction_mode` is `false`.

```
{
            "api_name": "rest_api",
            "host_name": "*",
            "url": "/",
            "api_type": "regular",
            "creation_date": "Fri Apr 05 05:41:00 UTC 2019",
            "servers": 2,
            "protocol": "http",
            "cookie": "",
            "token": false,
             "training_started_at": "n/a", "training_duration": "n/a", "prediction_mode": false}
```

Manage attack blocking

API Security Enforcer (ASE) and API Behavioral Security (ABS) work in tandem to detect and block attacks.

ASE detects attacks in real-time, blocks the hacker, and reports attack information to ABS. ABS AI Engine uses behavioral analysis to look for advanced attacks. Attack management is done in both ABS and ASE.

In ABS, you can:

- List active, expired or a consolidated list of active and expired client identifiers for a specific time period. For more information see, ABS deny list reporting.
- Delete specific client identifiers from ABS deny list or bulk delete a type of client identifier using ABS REST API. For more information, see Deleting individual client identifiers and Using the bulk delete option for client identifiers.
- Enable or disable a specific attack ID. When you disable an attack ID, ABS stops reporting attacks across all client identifiers for that attack ID. For more information, see Enabling or disabling attack IDs.

Copyright © 2025 Ping Identity Corporation

- Configure the time-to-live (TTL) for each client identifier type. The TTL time applies to all the detected attacks for that client identifier. For more information, see TTL for client identifiers in ABS.

In ASE, you can:

- Manually add or delete entries from allow list and deny list
- Enable or disable automatic blocking of ABS detected attack types
- Enable or disable ASE detected real-time attacks. ASE detects real time attacks only in an inline deployment.

For more information see, Attack management in ASE

ABS deny list reporting

API Behavioral Security (ABS) provides `attacklist` REST application programming interface (API) to complete the following two operations:

- List the various client identifiers (API Key, OAuth token, Username, Cookie, and Internet Protocol (IP) address) which are related to probable attack
- Delete the client identifiers which may be a cause of false positive

Reporting active and expired client identifiers

API Behavioral Security (ABS) provides an `attacklist` REST API with GET method to list active attacks in the system, expired attacks, and consolidated (active and expired) attacks together.

About this task

The list of detected client identifiers depends on the TTL set for the client identifiers. The attack list reports the detected client identifiers (active or expired) for the queried period. The time-period is part of the API query parameter.
`URL: /v4/abs/attacklist`

Steps

- To report active detected attacks, use the following REST API URL to report the active client identifiers: `/v4/abs/attacklist?earlier_date=<>&later_date=<>&status=active`

  The API lists the active client identifiers for a time-period between `earlier_date` and `later_date`. PingIntelligence ASE fetches the active client identifiers list from ABS for blocking the clients.

- To report expired detected attacks, use the following REST API URL to report the expired client identifiers: `/v4/abs/attacklist?earlier_date=<>&later_date=<>&status=expired`

  The API lists the expired client identifiers for a time-period between `earlier_date` and `later_date`. The expiry of detected attacks in the system depends on the configured TTL.

- To report consolidated (active and expired) detected attacks, use the following REST API URL to report the consolidated client identifiers attacks: `/v4/abs/attacklist?earlier_date=<>&later_date=<>`

  The API lists all the client identifiers for a time-period between `earlier_date` and `later_date`.

Copyright © 2025 Ping Identity Corporation

Deleting individual client identifiers

You can delete active client identifiers.

About this task

The API requires only the message body with a client identifier in their respective sections, to delete active client identifiers. The API checks if the client identifier is present in the blocklist or not before deleting. If you provide a client identifier which is not part of the blocklist, the API ignores such client identifiers.

Steps

- Use the **`attacklist`** API with PUT method to delete the client identifiers:
  - `URL: /v4/abs/attacklist`
  - `Method: PUT`

    > **Note**
    >
    > You can provide only specific section of a client identifier in the message body. For example, if you only want to delete specific usernames, then provide only the username section in the message body. Make sure that the `JavaScript Object Notation (JSON)` file is well-formed.

    Example:

    The following is a sample message body for the **`attacklist`** API to delete client identifiers:

Copyright © 2025 Ping Identity Corporation

```
{
        "ips": [
            "192.168.4.10",
            "10.10.10.73",
            "10.1.1.4",
            "10.9.8.7"
        ],
        "cookies": {
            "PHPSESSIONID": [
            "Cookie1",
            "Cookie2"
            ],
        "JSESSIONID": [
            "Cookie3",
            "AnyCookie",
            "Cookie4"

        },
        "oauth_tokens": [
            "Token1",
            "Token2",
            "Token3"
        ],
        "api_keys": [
            "type2_api_key",
            "api_key_1",
            "api_key_2",
         ],
        "usernames": [
            "username1",
            "username2",
            "username3",
         ]
}
```

The following is a sample message body showing the client identifiers that were deleted:

Copyright © 2025 Ping Identity Corporation

```
{
  "message": "Success: The following attacks have been removed:",
  "date": "Thu Jun 09 03:39:12 UTC 2019",
  "attacklist": {
    "ips": [
            "192.168.4.10",
            "10.10.10.73",
            "10.1.1.4",
            "10.9.8.7"
    ],
    "cookies": {
      "PHPSESSIONID": [
            "Cookie1",
            "Cookie2"
      ],
      "JSESSIONID": [
            "Cookie3",
            "AnyCookie",
            "Cookie4"
      ]
    },
    "oauth_tokens": [
            "Token1",
            "Token2",
            "Token3"
    ],
    "api_keys": [
            "type2_api_key",
            "api_key_1",
            "api_key_2",
    ],
    "usernames": [
            "username1",
            "username2",
            "username3",
    ]
  }
}
```


Using the bulk delete option for client identifiers

You can use the bulk delete option to clear large numbers of false positive client identifiers.

About this task

> **Note**
>
> You can also use the bulk delete option to clear the blocklist in case of a reset.

Steps

1. To bulk delete client identifiers, use the ABS `attacklist` REST API with the DELETE method:
   - `URL: /v4/abs/attacklist`
   - `Method: DELETE`

Copyright © 2025 Ping Identity Corporation

2. To bulk delete all the entries of a client identifier or all client identifiers, configure the body of the `attacklist`.

   Example:

   The following is an example of the API request:

```
{
        delete_all: false,
        delete_all_ips: true,
        delete_all_cookies: true,
        delete_all_oauth_tokens: false,
        delete_all_api_keys: true,
        delete_all_usernames: false,
}
```

> **Note**
>
> In the sample request body, the `attacklist` API deletes all entries for IP, cookies, and API keys. If, in the next time interval, the AI engine flags the same client identifiers, the blocklist is populated again.

3. To permanently stop a false positive from being reported, tune the thresholds using the PingIntelligence Web GUI for the specific client identifier.

   The following table describes the options.

Option	Description
`delete_all`	This option overrides all the other configured options in the message body. If it is set to `true`, all the client identifiers are deleted irrespective of what their individual configuration is. Set it to `false`, if you want to exercise other options.
`delete_all_ips`	Set it true to delete all the IP addresses across all attack types from the blocklist.
`delete_all_cookies`	Set it true to delete all the cookies across all attack types from the blocklist.
`delete_all_oauth_tokens`	Set it true to delete all the OAuth token across all attack types from the blocklist.
`delete_all_api_keys`	Set it true to delete all the API Keys across all attack types from the blocklist.
`delete_all_usernames`	Set it true to delete all the usernames across all attack types from the blocklist.

Copyright © 2025 Ping Identity Corporation

Enabling or disabling attack IDs

You can enable or disable one or more than one attack type using API Behavioral Security (ABS) `attackstatus` REST application programming interface (API) with the PUT method.

About this task

The AI engine keeps updating the thresholds in the background, even when you disable an attack ID. Calculating the thresholds in the background allows ABS to report attacks if you enable an attack ID in the future.

If you have disabled an attack while the AI engine is processing the log data, ABS may still report attacks for a few minutes. The attack IDs would be disabled when the next batch of access log files are processed. When you enable an attack from the disabled state, ABS takes a few minutes to report the API attacks.

URL: `/v4/abs/attackstatus`

Method: `PUT`

The following attack IDs cannot be disabled from ABS as these are real-time attacks reported by ASE:

1. Attack ID 13: API DDoS Attack Type 2
2. Attack ID 100: Decoy Attack. This attack ID can be disabled from ASE.
3. Attack ID 101: Invalid API Activity. This attack ID can be disabled from ASE.

To enable or disable an attack ID:

Steps

1. Run the `attackstatus` REST API with the GET method to fetch the current status of an attack ID.

   The output is divided into two sections, enabled and disabled, along with the time when an attack ID was enabled or disabled.

   Result:

   The following is a snippet of the response:

Copyright © 2025 Ping Identity Corporation

```
“attack_status”: {
  “enabled” : [
         {
            "attack_id" : 1,
            "attack_name" : "Data Exfiltration Attack Type 1",
            "enabled_time" : "Thu Aug 22 12:56:39:158 2019"
         },
         {
            "attack_id" : 2,
            "attack_name" : "Single Client Login Attack Type 1",
            "enabled_time" : "Thu Aug 22 12:56:39:158 2019"
         },
        {
            "attack_id" : 4,
            "attack_name" : "Stolen Token Attack Type 1",
            "enabled_time" : "Thu Aug 22 12:56:39:158 2019"
        }
             ],
“disabled” : [
         {
            "attack_id" : 3,
            "attack_name" : "Data Exfiltration Attack Type 1",
            "disabled_time" : "Thu Aug 22 12:56:39:158 2019"
         },
         {
            "attack_id" : 5,
            "attack_name" : "Single Client Login Attack Type 1",
            "disabled_time" : "Thu Aug 22 12:56:39:158 2019"
         }
                ]
}
```


> **Note**
>
> Attack IDs 13, 100, and 101 are always displayed as enabled in the response.

2. Use the `attackstatus` REST API with PUT method to enable or disable the attack IDs.

    1. Enter the `attack_id` and `action`.

        Result:

        The following is a sample `body` of the PUT request:

Copyright © 2025 Ping Identity Corporation

```
{
    “attacks”:[
  {
    "attack_id": “1”,
    "action": “disable”
  },
  {
    "attack_id": “2”,
    "action": “enable”
  },
  {
    "attack_id": “13”,
    "action": “disable”
  },
{
    "attack_id": “100”,
    "action": “disable”
  },
 {
    "attack_id": “101”,
    "action": “disable”
  }
]
}
```

The following is a sample response:

+

Copyright © 2025 Ping Identity Corporation

```
{
   "attack_status": [
       {
           "attack_id": "1",
           "attack_name": "Data Exfiltration Attack Type 1",
           "status": "Attack ID disabled successfully"
       },
       {
           "attack_id": "2",
           "attack_name": "Single Client Login Attack Type 1",
           "status": "Attack ID is already enabled"
       },
       {
           "attack_id": "13",
           "attack_name": "API DDoS Attack Type 2",
           "status": "Attack ID cannot be disabled. For more information, refer to PingIntelligence
documentation."
       },
       {
           "attack_id": "100",
           "attack_name": "Decoy Attack",
           "status": "Attack ID cannot be disabled. For more information, refer to PingIntelligence
documentation."
       },
       {
           "attack_id": "101",
           "attack_name": "Invalid API Activity",
           "status": "Attack ID cannot be disabled. For more information, refer to PingIntelligence
documentation."
       }

   ]
}
```


TTL for client identifiers in ABS

The API Behavioral Security (ABS) AI Engine deny list supports configuring the length of time that a client identifier type (username, OAuth token, API Key, cookie, and Internet Protocol (IP) address) remains on the deny list.

Each client identifier type can be configured with a different value in minutes. The default value of zero minutes means that the AI engine will not remove any client identifiers from the deny list unless the TTL value is changed.

You can change the default value of TTL by using the `admin` ABS REST application programming interface (API) which supports configuring a different TTL in minutes for each client identifier. Following are the recommended steps to managing client identifier TTL:

1. Use the ABS `admin` REST API to fetch the current TTL values.
2. Use the PUT method with the ABS `admin` REST API to configure the TTL.

When you update the TTL value, it applies to the client identifiers in the deny list that the AI engine identified from that time onwards. For example, you set initial TTL of 120 minutes at 6 a.m. for 100 client identifiers in the deny list, then the list will exist till 8 a.m.. Now, if you change the TTL at 7 AM to 30 minutes, then the initial list of 100 client identifier will still exist till 8 a.m.. The new 30 minute TTL will apply to the client identifiers reported from 7 a.m. onwards.

Copyright © 2025 Ping Identity Corporation

Use the `admin` API to fetch the current TTL of the client identifiers:

`https://<ip>:<port>/v4/abs/admin`. Following is a sample output displaying the current TTL values:

Copyright © 2025 Ping Identity Corporation

```
{
        "company": "ping identity",
        "name": "api_admin",
        "description": "This report contains status information on all APIs, ABS clusters,
      and ASE logs",
      "license_info": {
        "tier": "Subscription",
        "expiry": "Wed Jan 15 00:00:00 UTC 2020",
        "max_transactions_per_month": 1000000000,
        "current_month_transactions": 98723545,
        "max_transactions_exceeded": false,
        "expired": false
    },
    "across_api_prediction_mode": true,
    "api_discovery": {
        "subpath_length": "1",
        "status": true
        "apis": [
        {
        "api_name": "app",
        "host_name": "*",
        "url": "/atm_app_oauth",
        "api_type": "decoy-incontext",
        "creation_date": "Thu Dec 26 09:51:10 UTC 2019",
        "servers": 0,
        "protocol": "http",
        "cookie": "",
        "token": true,
        "training_started_at": "Thu Dec 26 09:52:29 UTC 2019",
        "training_duration": "1 hour",
        "prediction_mode": true,
        "apikey_header": "",
        "apikey_qs": ""
        }
        ],
        "abs_cluster": {
        "abs_nodes": [
        {
                "node_ip": "172.17.0.1",
                "os": "DISTRIB_ID=Ubuntu - ",
                "cpu": "4",
                "memory": "7.8G",
                "filesystem": "19%",
                "bootup_date": "Wed Dec 25 15:01:06 UTC 2019"
        }
        ],
        "mongodb_nodes": [
        {
                "node_ip": "172.17.0.1",
                "status": "up"
        }
        ]
        },
        "ase_logs": [
        {
        "ase_node": "8f9d07c5-c5c4-43c3-97be-9672c7fd2986",
        "last_connected": "Thu Dec 26 10:51:13 UTC 2019",
        "logs": {
                "start_time": "Thu Dec 26 09:51:14 UTC 2019",
                "end_time": "Thu Dec 26 10:51:13 UTC 2019",
```


Copyright © 2025 Ping Identity Corporation

```
                "gzip_size": "429.96KB"
        }
        }
        ],
        "percentage_diskusage_limit": "80%",
        "scale_config": {
        "scale_up": {
        "cpu_threshold": "70%",
        "cpu_monitor_interval": "30 minutes",
        "memory_threshold": "70%",
        "memory_monitor_interval": "30 minutes",
        "disk_threshold": "70%",
        "disk_monitor_interval": "30 minutes"
        },
        "scale_down": {
        "cpu_threshold": "10%",
        "cpu_monitor_interval": "300 minutes",
        "memory_threshold": "10%",
        "memory_monitor_interval": "300 minutes",
        "disk_threshold": "10%",
        "disk_monitor_interval": "300 minutes"
        }
        },
        "attack_ttl": {
        "ids": [
        {
                "id": "ip",
                "ttl": 0
        },
        {
                "id": "cookie",
                "ttl": 0
        },
        {
                "id": "access_token",
                "ttl": 0
        },
        {
                "id": "api_key",
                "ttl": 0
        },
        {
                "id": "username",
                "ttl": 0
        }
        ]
        }
}
```


Use the PUT method with `admin` REST API to configure the TTL in minutes:

`URL: https://<ip>:<port>/v4/abs/admin`

`Method: PUT`

Body:

Copyright © 2025 Ping Identity Corporation

```
{
 "ids" : [
 {
 "id" : "ip",
 "ttl" : 10
 },
 {
 "id" : "cookie",
 "ttl" : 10
 },
 {
 "id" : "access_token",
 "ttl" : 10
 },
 {
 "id" : "api_key",
 "ttl" : 10
 },
 {
 "id" : "username",
 "ttl" : 10
 }
 ]
}
```

Response:

```
{
        "message": "TTL updated successfully",
        "date": "Thu Dec 26 10:59:40 UTC 2019"
}
```

To verify the new TTL values, rerun the ABS `admin` REST API with the GET method.

Configuring automated ASE attack blocking

When the AI Engine detects an attack, it adds an entry to its deny list, which consists of usernames, tokens, application programming interface (API) Keys, cookies, and Internet Protocol (IP) addresses of clients that were detected executing attacks.

About this task

If blocking is enabled for the API, the deny list is automatically sent to API Security Enforcer (ASE) nodes, which blocks the client's future access using the identifiers on the list.

Steps

- To activate API Behavioral Security (ABS) log processing, run the following ASE command:

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs
  ```


Copyright © 2025 Ping Identity Corporation

> **Note**
>
> After log processing is enabled, ASE sends log data to ABS which processes the log data to look for attacks and generate reports.

- To activate automatic client blocking, run the following ASE command:

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs_attack
  ```

> **Note**
>
> ABS generates a list of clients which are suspected of executing attacks. ABS can be configured to automatically send the attack list to ASE which blocks client access. By default, automatic blocking is inactive.

- To disable automatic sending of ABS attack lists to ASE, run the following ASE command:

  Example:

  ```
  /opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs_attack
  ```

Attack management in ASE

In API Security Enforcer (ASE), you manage detected attacks through both allow list and deny list.

Client identifiers in deny list are blocked by ASE while those in the allow list are never blocked. You can also choose to block or allow a client identifier at application programming interface (API) level by configuring the individual API JavaScript Object Notation (JSON).

Allow list

List of safe Internet Protocol (IP) addresses, cookies, OAuth2 Tokens, API keys, or usernames that will not be blocked by ASE.The list is manually created using ASE CLI commands.

Deny list

List of bad IP addresses, cookies, OAuth2 Tokens, API keys, or usernames that are always blocked by ASE.The list consists of entries from one or more of the following sources:

- API Behavioral Security (ABS) detected clients suspected of executing attacks (for example, data exfiltration).
- ASE detected clients suspected of executing attacks (for example, invalid method, decoy API accessed). These attacks are reported to ABS and become part of ABS deny list also after further AI processing.
- List of bad client identifiers manually added using ASE CLI

Copyright © 2025 Ping Identity Corporation

Managing the ASE allow list

To manage API Security Enforcer (ASE) operations for OAuth2 tokens, cookies, Internet Protocol (IP) addresses, username, and application programming interface (API) keys on an allow list.

About this task

To manage the ASE allow list:

Steps

- To add an IP address to an allow list, run the `add_whitelist` command with the `ip` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist ip 10.10.10.10
ip 10.10.10.10 added to whitelist
```

- To add a cookie to an allow list, run the `add_whitelist` command with the `cookie` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist cookie JSESSIONID cookie_1.4
cookie JSESSIONID cookie_1.4 added to whitelist
```

- To add a token to an allow list, run the `add_whitelist` with the `token` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist token token1.4
token token1.4 added to whitelist
```

- To add an API key to an allow list, run the `add_whitelist` command with the `api_key` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist api_key X-API-KEY key_1.4
api_key X-API-KEY key_1.4 added to whitelist
```

- To add a username to an allow list, run the `add_whitelist` command with the `username` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist username user1
username user1 added to whitelist
```

- To view an allow list, run the `view_whitelist` command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_whitelist
Whitelist
1) type : ip, value : 1.1.1.1
2) type : cookie, name : JSESSIONID, value : cookie_1.1
3) type : token, value : token1.3
4) type : api_key, name : X-API-KEY, value : key_1.4
```

- To delete an entry from an allow list, run the `delete_whitelist` command:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist ip 4.4.4.4
ip 4.4.4.4 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist cookie JSESSIONID cookie_1.1
cookie JSESSIONID cookie_1.1 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist token token1.1
token token1.1 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist api_key X-API-KEY key_1.4
api_key X-API-KEY key_1.4 deleted from whitelist

/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_whitelist username user1
username user1 deleted from whitelist
```

- To clear the allow list, run the `clear_whitelist` command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin clear_whitelist
This will delete all whitelist Attacks, Are you sure (y/n) : y
Whitelist cleared
```

Managing the ASE deny list

Manage API Security Enforcer (ASE) operations for OAuth2 tokens, cookies, Internet Protocol (IP) addresses, username, and application programming interface (API) keys on a deny list.

About this task

To manage the ASE deny list:

Steps

- To add an IP address to the deny list, run the `add_blacklist` command with the `ip` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist ip 1.1.1.1
ip 1.1.1.1 added to blacklist
```

- To add a cookie to a deny list, run the `add_blacklist` command with the `cookie` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist cookie JSESSIONID
ad233edqsd1d23redwefew
cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
```

- To add a token to a deny list, run the `add_blacklist` command with the `token` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token ad233edqsd1d23redwefew
token ad233edqsd1d23redwefew added to blacklist
```

Copyright © 2025 Ping Identity Corporation

- To add an API key to a deny list, run the `add_blacklist` command with the `api_key` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist api_key AccessKey
b31dfa4678b24aa5a2daa06aba1857d4
api_key AccessKey b31dfa4678b24aa5a2daa06aba1857d4 added to blacklist
```

- To add a username to a deny list, run the `add_black list` command with the `username` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist username user1
username user1 added to blacklist
```

- To view the entire deny list, run the `view_blacklist` command with the `all` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist all
Manual Blacklist
1) type : ip, value : 10.10.10.10
2) type : cookie, name : JSESSIONID, value : cookie_1.4
3) type : token, value : token1.4
4) type : api_key, name : X-API-KEY, value : key_1.4
Realtime Decoy Blacklist
1) type : ip, value : 4.4.4.4
Realtime Protocol Blacklist
1) type : token, value : token1.1
2) type : ip, value : 1.1.1.1
3) type : cookie, name : JSESSIONID, value : cookie_1.1
Realtime Method Blacklist
1) type : token, value : token1.3
2) type : ip, value : 3.3.3.3
3) type : cookie, name : JSESSIONID, value : cookie_1.3
Realtime Content-Type Blacklist
1) type : token, value : token1.2
2) type : ip, value : 2.2.2.2
3) type : cookie, name : JSESSIONID, value : cookie_1.2
```

- To view the deny list based on decoy IP addresses, run the `view_blacklist` command with the `decoy` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist decoy
Realtime Decoy Blacklist
1) type : ip, value : 4.4.4.4
```

- To view the deny list based on protocol violations, run the `view_blacklist` command with the `invalid_protocol` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist invalid_protocol
Realtime Protocol Blacklist
1) type : token, value : token1.1
2) type : ip, value : 1.1.1.1
3) type : cookie, name : JSESSIONID, value : cookie_1.1
```


Copyright © 2025 Ping Identity Corporation

- To view the deny list based on method violations, run the `view_blacklist` command with the `invalid_method` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist invalid_method
Realtime Method Blacklist
1) type : token, value : token1.3
2) type : ip, value : 3.3.3.3
3) type : cookie, name : JSESSIONID, value : cookie_1.3
```

- To view the deny list based on content-type violation, run the `view_blacklist` command with the `invalid_content_type` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist invalid_content_type
Realtime Content-Type Blacklist
1) type : token, value : token1.2
2) type : ip, value : 2.2.2.2
3) type : cookie, name : JSESSIONID, value : cookie_1.2
```

- To view API Behavioral Security (ABS) -detected attacks, run the `view_blacklist` command with the `abs_detected` option:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist abs_detected
No Blacklist
```

- To delete an entry from a deny list, run the `delete_blacklist` command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin delete_blacklist ip 1.1.1.1
ip 1.1.1.1 deleted from blacklist

./bin/cli.sh -u admin -p admin delete_blacklist cookie JSESSIONID avbry47wdfgd
cookie JSESSIONID avbry47wdfgd deleted from blacklist

./bin/cli.sh -u admin -p admin delete_blacklist token 58fcb0cb97c54afbb88c07a4f2d73c35
token 58fcb0cb97c54afbb88c07a4f2d73c35 deleted from blacklist
```

- To clear the deny list, run the `clear_blacklist` command:

```
./bin/cli.sh -u admin -p admin clear_blacklist
This will delete all blacklist Attacks, Are you sure (y/n) :y
Blacklist cleared
./bin/cli.sh -u admin -p admin clear_blacklist
This will delete all blacklist Attacks, Are you sure (y/n) :n
Action canceled
```

Copyright © 2025 Ping Identity Corporation

> **Warning**
>
> When clearing the deny list, make sure that Real-time Detected attacks for inline ASE attacks and ABS-detected attacks are disabled. If not disabled, the deny list gets populated again as both ASE and ABS are continuously detecting attacks.

Enabling API blocking in ASE

You can configure API Security Enforcer (ASE) to selectively block on a per application programming interface (API) basis by configuring an API JavaScript Object Notation (JSON) file parameter.

About this task

To enable per API blocking for each API:

Steps

- **Set the `enable_blocking` parameter to `true` in the API JSON file.**

  Example:

```
api_metadata": {
 "protocol": "http",
 "url": "/",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "enable_blocking": true,
 "login_url": "",
 "api_mapping": {
 "internal_url": ""
 },
```

> **Note**
>
> If per API blocking is disabled, API Behavioral Security (ABS) still detects the suspected attacks for that specific API, however, ASE does not block them. ASE will continue to block the suspected attacks on other APIs with the `enable_blocking` set to `true`.

Attack reporting

Attack reports provide information about the suspected attacks on each application programming interface (API).

The API Behavioral Security (ABS) Attack API provides reports by specifying the `type_id` (see descriptions in Attack Types REST and WebSocket APIs) and receiving attack details including time frame, client identifier, and an attack code (seeChanging Attack Thresholds for an explanation of attack codes). The format of the ABS `attack` API is:

Copyright © 2025 Ping Identity Corporation

https://<*hostname*>:<*port*>/v4/abs/later_date<>&earlier_date<>&api=<*api_name*>type=type_id

The hostname and port correspond to the host ABS machine.

Understanding the API report parameters

Here is a brief description of the information available in the attack reports. Not all items are included in each of the reports. Please refer toABS external REST APIs for detailed information in each report.

- `attack_type`: Name of the attack type (for example, data exfiltration, stolen cookie)
- `description`: Description of the attack.
- `earlier_date`: A date which is past in time. For example, if the query range is between March 12 and March 14, then the earlier date would be March 12.
- `later_date`: A date which is more recent in time. For example, if the query range is between March 12 and March 14, then the later date would be March 14.
- `api_name`: The name of the API for which report is displayed.
- `access_time`: The time that the hacker accessed the API
- `attack_code`: Code for the variables and thresholds used to detect attacks. For example, attack_code": "varA(Tx, 25) signifies that the attack was triggered because variable A with a value of 25 exceeded the Tx threshold. Current threshold values can be checked using the Threshold API.
- `ddos_info`: The `ddos_info` field provides a pointer to detailed information in the MongoDB system – for example, a list of IPs that were active during a DDoS attack (note: only included in DDoS reports). The data is accessible in the `log in_dos` collection in `abs_data` database. To access the data, enter the following in your MongoDB command line:

```
>use abs_data
>db.login_dos.find({end_time:'Tue Mar 21 22:25:36:144 2017'},{'ips':1}).pretty()
```

Use the `end_time` in the query to see the participating IPs.

The following pages provide examples of API JavaScript Object Notation (JSON) attack reports for Data Exfiltration, Stolen Cookie, and Multi-Client Login Attack.

> **Note**
>
> You can use the Admin user or the restricted user to access the API reports. For the Admin user, the cookie, token or the API key is not obfuscated.

Consolidated result of attack types

To view all attack types on a given application programming interface (API) in a single, consolidated report, use the API Behavioral Security (ABS) Attack API.

Attack ID 0 gives all the attacks on a single API or across APIs based on the REST API query parameters.

Copyright © 2025 Ping Identity Corporation

Consolidated attack report for an API

Steps

- Use API URL with attack ID as 0 to access all the attacks for a specific API:

  https://<*ABS_IP:port*>/v4/abs/attack?later_date=yyyy-mm-ddThh:mm&later_date=yyyy-mm-ddThh:mm&api=<*api_name*>&type=<*type_id*>

  Example:

  https://192.168.11.166:8080/v4/abs/attack?later_date=2018-12-31T18:00&later_date=2018-10-25T13:30&api=shop&type=0

  1. You can further select a client identifier (Internet Protocol (IP), cookie, or a token) and carry out IP, cookie, or token forensics using the Forensic API.

     Example:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "attack_type": "Data Exfiltration Attack",
 "cookie": "JSESSIONID",
 "description": "Client (IP or Cookie) extracting an abnormal amount of data for given API",
 "earlier_date": "Tue Jan 02 16:00:00:000 2018",
 "later_date": "Mon Jan 01 18:00:00:000 2018",
 "api_name": "shop",
 "cookies": [
 {
 "cookie": "extreme_client_activity_500_request",
 "details": [
 {
 "access_time": "Fri Jan 12 08:44:39:086 2018",
 "attack_code": "varA(Tx, 26)",
 "attack_deviation": "varA(700%)"
 },
 {
 "access_time": "Fri Jan 12 09:18:34:087 2018",
 "attack_code": "varA(Tx, 25)",
 "attack_deviation": "varA(700%)"
 }
 ]
 },

 {
 "company": "ping identity",
 "attack_type": "API Probing Replay Attack",
 "cookie": "JSESSIONID",
 "description": "Client (IP or Cookie) probing or trying different parameter values to breach
 the API service for given API",
 "earlier_date": "Tue Jan 02 16:00:00:000 2018",
 "later_date": "Mon Jan 01 18:00:00:000 2018",
 "api_name": "shop",
 "cookies": [
 {
 "cookie": "api_dos_attack_type_1_shop_50_percent_error",
 "details": [
 {
 "access_time": "Fri Jan 12 08:39:56:896 2018",
 "attack_code": "varA(Tx, 47)",
 "attack_deviation": "varA(700%)"
 },
 {
 "access_time": "Fri Jan 12 09:18:34:087 2018",
 "attack_code": "varA(Tx, 47)",
 "attack_deviation": "varA(700%)"
 }
 },
 },
}
```


Consolidated attack report across API

Steps

- Use the following ABS REST API to access all the attack types:

Copyright © 2025 Ping Identity Corporation

https://<*ABS_IP:port*>/v4/abs/attack?later_date=yyyy-mm-ddThh:mm&later_date=yyyy-mm-ddThh:mm&type=<*type_id*>.

Example:

https://192.168.11.166:8080/v4/abs/attack?later_date=2018-12-31T18:00&later_date=2018-10-25T13:30&type=0

1. **You can further select a client identifier (IP, cookie, or a token) and carry out IP, cookie, or token forensics using the Forensic API.**

   Example:

Copyright © 2025 Ping Identity Corporation

```
[
    {
        "company": "ping identity",
        "attack_type": "Stolen Token Attack Type 2",
        "name": "api_attack_type",
        "description": "Client (Token) reusing cookies to deceive application services.",
        "earlier_date": "Thu Oct 25 13:30:00:000 2018",
        "later_date": "Mon Dec 31 18:00:00:000 2018",
        "api_name": "all",
        "access_tokens": [
            {
                "access_token": "SYU4R2ZZN1IDYI0L",
                "details": [
                    {
                        "access_time": "Tue Nov 27 11:10:00:000 2018",
                        "attack_code": "varA(Tn, 3)",
                        "attack_deviation": "varA(700%)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 3)",
                        "attack_deviation": "varA(700%)"
                    },
                    {
                        "access_time": "Tue Nov 27 16:10:00:000 2018",
                        "attack_code": "varA(Tn, 2)",
                        "attack_deviation": "varA(700%)"
                    }
                ]
            },
            {
                "access_token": "CT27QTP01K6ZW2AK",
                "details": [
                    {
                        "access_time": "Tue Nov 27 10:50:00:000 2018",
                        "attack_code": "varA(Tn, 2)",
                        "attack_deviation": "varA(700%)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:10:00:000 2018",
                        "attack_code": "varA(Tn, 4)",
                        "attack_deviation": "varA(700%)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 5)",
                        "attack_deviation": "varA(700%)"
                    }
                ]
            },

            {
                "ip": "100.64.7.124",
                "details": [
```


Copyright © 2025 Ping Identity Corporation

```
                    {
                        "access_time": "Tue Nov 27 11:20:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)",
                        "attack_deviation": "varA(700%)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:30:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)",
                        "attack_deviation": "varA(700%)"
                    }
                ]
            },

            {
                "ip": "100.64.10.18",
                "details": [
                    {
                        "access_time": "Tue Nov 27 11:10:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)",
                        "attack_deviation": "varA(700%)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)",
                        "attack_deviation": "varA(700%)"
                    }
                ]
            }
        ]
    }
]
```


Real-time Detected attacks for inline ASE

API Security Enforcer (ASE) supports real time attack detection and blocking for:

- API Pattern Enforcement – validate traffic to ensure it is consistent with the application programming interface (API) definition
- API Deception – blocks hackers probing a Decoy API

In real-time, ASE blocks hackers which violate pattern enforcement or probe decoy APIs. Hacker information is reported to API Behavioral Security (ABS) which generates ASE detected attack reports (type ID 101). Use the following ABS REST API to view the report:

https://192.168.11.138:8080/v4/abs/attack?later_date=2018-07-16&earlier_date=2018-07-16&api=atmapp&type=101

Real-time ASE detected attack based on OAuth2 token activity

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "attack_type": "Invalid API Activity",
 "name": "api_attack_type",
 "description": "Clients using invalid method/protocol/content-type",
 "earlier_date": "Thu Jan 25 18:00:00:000 2018",
 "later_date": "Fri Dec 28 18:00:00:000 2018",
 "api_name": "atm_app_oauth",
 "ips": [],
 "cookies": [],
 "access_tokens": [
 {
 "access_token": "token_protocol",
 "details": [
 {
 "access_time": "Fri Jan 26 20:58:04:770 2018",
 "attack_code": "protocol"
 },
 {
 "access_time": "Fri Jan 26 21:16:17:851 2018",
 "attack_code": "protocol"
 }
 ]
 },
 {
 "access_token": "token_method",
 "details": [
 {
 "access_time": "Fri Jan 26 20:58:04:819 2018",
 "attack_code": "method"
 },
 {
 "access_time": "Fri Jan 26 21:16:17:903 2018",
 "attack_code": "method"
 }
 ]
 },
 {
 "access_token": "token_contenttype",
 "details": [
 {
 "access_time": "Fri Jan 26 20:58:04:819 2018",
 "attack_code": "content_type"
 },
 {
 "access_time": "Fri Jan 26 21:16:17:903 2018",
 "attack_code": "content_type"
 }
 ]
 }
 ]
}
```


Real-time ASE detected attack based on pattern enforcement violation

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "attack_type": "Invalid API Activity",
 "cookie": "JSESSIONID",
 "name": "api_attack_type",
 "description": "Clients using invalid method/protocol/content-type",
 "earlier_date": "Thu Jan 25 18:00:00:000 2018",
 "later_date": "Fri Dec 28 18:00:00:000 2018",
 "api_name": "atm_app_public",
 "ips": [],
 "cookies": [
 {
 "cookie": "session_contenttype1",
 "details": [
 {
 "access_time": "Fri Jan 26 21:17:10:662 2018",
 "attack_code": "content_type"
 }
 ]
 },
 {
 "cookie": "session_method",
 "details": [
 {
 "access_time": "Fri Jan 26 20:58:06:656 2018",
 "attack_code": "method"
 },
 {
 "access_time": "Fri Jan 26 21:17:10:662 2018",
 "attack_code": "method"
 }
 ]
 },
 {
 "cookie": "session_contenttype",
 "details": [
 {
 "access_time": "Fri Jan 26 20:58:06:656 2018",
 "attack_code": "content_type"
 },
 {
 "access_time": "Fri Jan 26 21:17:10:662 2018",
 "attack_code": "content_type"
 }
 ]
 },
 {
 "cookie": "session_protocol",
 "details": [
 {
 "access_time": "Fri Jan 26 20:58:04:873 2018",
 "attack_code": "protocol"
 },
 {
 "access_time": "Fri Jan 26 21:16:47:314 2018",
 "attack_code": "protocol"
 }
 ]
 },
 {
```


Copyright © 2025 Ping Identity Corporation

```
  "cookie": "session_method1",
  "details": [
  {
  "access_time": "Fri Jan 26 21:17:10:662 2018",
  "attack_code": "method"
  }
  ]
  },
  {
  "cookie": "session_protocol1",
  "details": [
  {
  "access_time": "Fri Jan 26 21:16:47:314 2018",
  "attack_code": "protocol"
  }
  ]
  }
  ],
  "access_tokens": []
 }
```


Enabling ASE detected attacks

Steps

- To enable real-time ASE detected attacks, run the following command on the ASE command line:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_ase_detected_attack
ASE Detected Attack is now enabled
```

Disabling ASE detected attacks

Steps

- To disable real-time ASE detected attacks, run the following command on the ASE command line:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_ase_detected_attack
ASE Detected Attack is now disabled
```

> **Note**
>
> When you disable ASE detected attacks, the attacks are deleted from the deny list.

Anomalous activity reporting

The Anomaly application programming interface (API) provides detailed reporting on anomalous activity associated with a specified API.

The types of anomalies detected include:

- Anomalies for each API Behavioral Security (ABS) attack type – activity which has the characteristics of one of the attack types (for example, API Memory Attack) ,but does not meet the threshold of an attack.

Copyright © 2025 Ping Identity Corporation

- Irregular Uniform Resource Locator (URL) – suspicious URL traffic.
- Anomalous request activity including injection attacks, overflow attacks, and system commands.

This report detects leading indicators of attacks on API services and is reviewed to observe trends.

Here is an snippet from an Anomaly API JavaScript Object Notation (JSON) report for a cookie-based API:


```
{
 "company": "ping identity",
 "name": "api_anomalies",
 "description": " This report contains information on anomalous activity on the specified
 API",
 "later_date": "Tue Jan 14 18:00:00:000 2018",
 "earlier_date": "Sun Jan 12 18:00:00:000 2018",
 "api_name": "shop",
 "anomalies_summary": {
 "api_url": "shopapi",
 "total_anomalies": 14,
 "most_suspicious_ips": [],
 "most_suspicious_anomalies_urls": []
 },
 "anomalies_details": {
 "url_anomalies": {
 "suspicious_sessions": [],
 "suspicious_requests": []
 },
 "ioc_anomalies": [
 {
 "anomaly_type": "API Memory Attack Type 2",
 "cookies": [
 {
 "cookie": "AMAT_2_H",
 "access_time": [
 "Mon Jan 13 01:01:33:589 2018"
 ]
 },
 {
 "cookie": "AMAT_2_H",
 "access_time": [
 "Mon Jan 13 01:01:33:589 2018"
 ]
 }
 ]
 },
```


Deception and decoy API

API Security Enforcer (ASE) supports configuration of decoy application programming interface (API), either for the in-context or out-of-context mode.

API Deception

If a client accesses an ASE decoy API and later tries to access a legitimate API, ASE drops the connection and blocks the client from accessing any non-decoy APIs. ASE Admin Guide provides more information on API Deception Environments.

Copyright © 2025 Ping Identity Corporation

Report ASE real-time decoy attack detection

ASE sends information about clients accessing decoy APIs to API Behavioral Security (ABS) which does further analysis and generates an API Deception report with type ID 100.

Here is an example ABS REST API to generate an API Deception report:

https://192.168.11.138:8080/v4/abs/attack?later_date=2018-07-16&earlier_date=2018-07-16&api=atmapp&type=100

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "attack_type": "Decoy Attack",
 "name": "api_attack_type",
 "description": "Clients accessing decoy APIs",
 "earlier_date": "Mon Jan 01 12:00:00:000 2018",
 "later_date": "Mon Dec 31 02:28:00:000 2018",
 "api_name": "atmapp",
 "ips": [
 {
 "ip": "100.64.38.140",
 "details": [
 {
 "access_time": "Sun Jan 28 19:59:29:395 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 19:59:29:395 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
 }
 ]
 },
 {
 "ip": "100.64.38.144",
 "details": [
 {
 "access_time": "Sun Jan 28 19:59:29:395 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 19:59:29:395 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
 },
 {
 "access_time": "Sun Jan 28 21:18:01:501 2018",
 "attack_code": "decoy"
```


Copyright © 2025 Ping Identity Corporation

```
    },
    {
    "access_time": "Sun Jan 28 21:18:01:501 2018",
    "attack_code": "decoy"
    }
    ]
    }
    ],
    "cookies": [],
    "access_tokens": []
  }
```


Decoy API

When decoy APIs are configured in ASE, then ABS generates decoy API reports with detailed information on all client access to decoy APIs including ASE detected violations.

Here is a decoy APIUniform Resource Locator (URL):

`<ABS_IP>:port/v4/abs/decoy?earlier_date<>&` `later_date<>`

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "decoy_api_metrics",
 "description": "This report contains detailed information on client access to each decoy API
 ",
 "later_date": "Tue Jan 11 18:00:00:000 2018",
 "earlier_date": "Tue Jan 11 17:50:00:000 2018",
 "api_name": "atmapp",
 "api_type": "decoy-incontext",
 "decoy_url": [
 "/atmapp/decoy"
 ],
 "summary": [
 {
 "decoy_url": "/atmapp/decoy",
 "unique_ip_count": 122,
 "total_requests": 240,
 "most_used_methods": {
 "GET": 88,
 "DELETE": 32,
 "ABDU": 32,
 "POST": 30,
 "PUT": 26
 },
 "most_used_ips": {
 "100.64.9.37": 4,
 "100.64.10.79": 4,
 },
 "most_used_devices": {
 "UBUNTU": 76,
 "MAC_OS_X": 69,
 },
 "most_used_content_types": {
 "UNKNOWN": 184,
 "multipart/form-data": 56
 }
 }
 ],
 "details": [
 {
 "decoy_url": "/atmapp/decoy",
 "source_ip": [
 {
 "ip": "100.64.31.183",
 "total_requests": 2,
 "method_count": {
 "GET": {
 "count": 2
 }
 },
 "url_count": {
 "/atmapp/decoy": 2
```


For a full report, see ABS external REST APIs**.**

Copyright © 2025 Ping Identity Corporation

Blocked connection reporting

API Behavioral Security (ABS) Blocked Connection REST application programming interface (API) reports all connections that are blocked by API Security Enforcer (ASE).

Two types of reports are provided:

- Blocked Connection Summary Report
- Blocked Connection Detail Report

The blocked connections are reported for the following categories:

- API routing
- DDoS flow control
- ABS detected attacks
- Custom blacklist
- Decoy attacks
- ASE detected attacks

Use the following ABS REST API for viewing the blocked connections report:

Blocked connection summary

URL: `<ABS_IP>:port/v4/abs/bc?earlier_date=<>T<hh:mm>&later_date=<>T<hh:mm>`

The following is a snippet of blocked connection summary report:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_blockedconnections",
 "description": " This report contains a summary of all API traffic blocked
  by ASE for the following types: api_not_found, host_header_not_found,
  backend_not_found, client_spike, server_spike, bytes_in_threshold,
  bytes_out_threshold, quota_threshold, customer_blacklist,
  abs_detected_attacks, ase_detected_attacks, decoy_detected_attacks",
 "earlier_date": "Thu Jan 18 13:00:00:000 2018",
 "later_date": "Thu Feb 22 18:00:00:000 2018",
 "api_name": "global",
 "total_blocked_connections": 21222,
 "api_not_found": 0,
 "host_header_not_found": 0,
 "backend_not_found": 3501,
 "client_spike": 237,
 "server_spike": 6179,
 "bytes_in_threshold": 5938,
 "bytes_out_threshold": 18,
 "quota_threshold": 0,
 "customer_blacklist": 0,
 "abs_detected_attacks": 4576,
 "ase_detected_attacks": 773,
 "decoy_detected_attacks": 0
```


Blocked Connection Details

URL: `<ABS_IP>:port/v4/abs/bc?later_date=<>T<hh:mm>&earlier_date=<> T<hh:mm>&details=true`

The following is a snippet of blocked connection details report:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_blockedconnections",
 "description": "This report contains details of all API traffic blocked by
  ASE for the following types: api_not_found, host_header_not_found,
  backend_not_found, client_spike, server_spike, bytes_in_threshold,
  bytes_out_threshold, quota_threshold, customer_blacklist,
  abs_detected_attacks,  ase_detected_attacks, decoy_detected_attacks,
 "earlier_date": "Thu Jan 18 13:00:00:000 2018",
 "later_date": "Thu Feb 22 18:00:00:000 2018",
 "api_blocked_connections": [
 {
 "category": "api_routing",
 "details": [
 {
 "source": "192.168.11.161",
 "type": "backend_not_found",
 "destination_api": "/v2/pet/55"
 },
 {
 "source": "192.168.11.161",
 "type": "backend_not_found",
 "destination_api": "/v2/store/inventory"
 }
 ]
 },
 {
 "category": "ddos_flowcontrol",
 "details": [
 {
 "source": "100.64.1.24",
 "type": "bytes_in_threshold",
 "destination_api": "/app/ws"
 },
 {
 "source": "100.64.3.213",
 "type": "protocol_violation",
 "destination_api": ""
 }
 ]
 },
 {
 "category": "abs_detected_attacks",
 "details": [
 {
 "source": "100.64.38.180",
 "type": "ioc_abs_ip_port",
 "destination_api": "/atmapp/zipcode"
 },
 {
 "source": "100.64.38.180",
 "type": "ioc_abs_ip_port",
 "destination_api": "/atmapp/zipcode"
 }
 ]
 },
 {
 "category": "customer_blacklist",
 "details": []
 },
```


Copyright © 2025 Ping Identity Corporation

```
  {
  "category": "decoy_detected_attacks",
  "details": []
  },
  {
  "category": "ase_detected_attacks",
  "details": [
  {
  "source": "100.64.8.252",
  "type": "protocol_violation",
  "destination_api": ""
  },
  {
  "source": "100.64.36.93",
  "type": "protocol_violation",
  "destination_api": ""
  }
  ]
  },
  ]
  }
  ]
}
```


API forensics reporting

API Behavioral Security (ABS) AI Engine provides in-depth information on the activities performed by a client including accessed Uniform Resource Locator (URL) , methods, attacks, etc.

The forensic report provides detailed information on the activity from an individual Token, Internet Protocol (IP) address, Cookie, API key, or Username.

> **Note**
>
> If API Security Enforcer (ASE) is deployed in sideband mode, then server field in the output shows the IP address as `0.0.0.0` . For ASE deployed in inline mode, the server field shows the IP address of the backend API server. For more information on ASE sideband mode, see the ASE Admin Guide.

Forensics on OAuth2 token

The OAuth2 token forensics report shows all activity associated with the specified token over a time period. Report information includes a detailed activity trail of accessed URLs, methods, and attacks.

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_token",
 "description": "This report contains a summary and detailed information on metrics,
  attacks and anomalies for the specified token across all APIs.",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "summary": {
 "total_requests": 6556,
 "total_attacks": 2,
 "total_anomalies": 0
 },
 "details": {
 "metrics": {
 "token": "token1",
 "total_requests": 6556,
 "ip_list": [
 {
 "ip": "127.0.0.1",
 "total_requests": 6556,
 "devices": {
 "UNKNOWN": 6556
 },
 "methods": {
 "DELETE": 472,
 "POST": 140,
 "GET": 1944,
 "PUT": 4000
 },
 "urls": {
 "/atm_app_oauth/delete200": 218,
 "/atm_app_oauth/get200": 850,
 "/atm_app_oauth/post400": 8,
 "/atm_app_oauth/post200": 62,
 "/atm_app_oauth/put400": 62,
 "/atm_app_oauth/get400": 122,
 "/atm_app_oauth/put200": 1938,
 "/atm_app_oauth/delete400": 18,
 "/2_atm_app_oauth/put200": 1938,
 "/2_atm_app_oauth/post200": 62,
 "/2_atm_app_oauth/delete200": 218,
 "/2_atm_app_oauth/delete400": 18,
 "/2_atm_app_oauth/put400": 62,
 "/2_atm_app_oauth/post400": 8,
 "/2_atm_app_oauth/get400": 122,
 "/2_atm_app_oauth/get200": 850
 },
 "apis": {
 "atm_app_oauth": 3278,
 "2_atm_app_oauth": 3278
 }
 }
 ]
 },
 "attack_types": {
 "API Memory Attack Type 1": [
 "atm_app_oauth",
 "2_atm_app_oauth"
 ],
 "Data Poisoning Attack": [
```


Copyright © 2025 Ping Identity Corporation

```
    "atm_app_oauth",
    "2_atm_app_oauth"
    ]
    },
    "anomaly_types": {}
    }
  }
```


Forensics on an IP address

The IP forensics report shows all activity associated with the specified IP address over a time period. Report information includes a detailed activity trail of accessed URLs, methods, and attacks.

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_ip",
 "description": "This report contains a summary and detailed information on
  metrics, attacks and anomalies for the specified ip across all APIs.",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "summary": {
 "total_requests": 8192,
 "total_attacks": 2,
 "total_anomalies": 1
 },
 "details": {
 "metrics": {
 "no_session": [
 {
 "start_time": "Thu Feb 15 14:04:17:959 2018",
 "end_time": "Thu Feb 15 14:05:59:263 2018",
 "total_requests": 4096,
 "source_ip": "4.1.1.1",
 "path": "/atm_app_private/get200",
 "methods": [
 "GET"
 ]
 },
 {
 "start_time": "Thu Feb 15 14:14:00:724 2018",
 "end_time": "Thu Feb 15 14:14:47:999 2018",
 "total_requests": 4096,
 "source_ip": "4.1.1.1",
 "path": "/2_atm_app_private/get200",
 "methods": [
 "GET"
 ]
 }
 ],
 "session": []
 },
 "attack_types": {
 "Data Exfiltration Attack": [
 "2_atm_app_private",
 "atm_app_private"
 ],
 "Extreme App Activity Attack": [
 "2_atm_app_private",
 "atm_app_private"
 ]
 },
 "anomaly_types": {
 "Extreme Client Activity Anomaly": [
 "2_atm_app_private"
 ]
 }
 }
}
```


Copyright © 2025 Ping Identity Corporation

Forensics on a Cookie

The Cookie forensics report includes all activity associated with the specified cookie over a time period. Report information includes a detailed activity trail of accessed URLs, methods, and attacks.

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_cookie",
 "description": "This report contains a summary and detailed information on all
  attacks, metrics, and anomalies for the specified cookie on the defined API.",
 "earlier_date": "Thu Jan 25 18:00:00:000 2018",
 "later_date": "Fri Dec 28 18:00:00:000 2018",
 "api_name": "atm_app_public",
 "summary": {
 "total_anomalies": 0,
 "total_requests": 1,
 "total_ioc": 2
 },
 "details": {
 "ioc_types": [
 "data_poisoning_attack",
 "api_memory_attack_type_1"
 ],
 "metrics": [
 {
 "session_id": "session_datapoisoining",
 "start_time": "Mon Jan 29 15:51:23:408 2018",
 "end_time": "Mon Jan 29 15:51:23:408 2018",
 "total_requests": 1,
 "source_ip": [
 {
 "ip": "127.0.0.1",
 "count": 1,
 "method": [
 "PUT"
 ]
 }
 ],
 "user_agent": [
 {
 "user_agent": "DOWNLOAD",
 "count": 1
 }
 ],
 "path_info": [
 {
 "path": "/atm_app_public/put200",
 "count": 1
 }
 ],
 "device": [
 {
 "device": "UNKNOWN",
 "count": 1
 }
 ],
 "server": [
 {
 "server": "127.0.0.1:3000",
 "count": 1
 }
 ]
```


Copyright © 2025 Ping Identity Corporation

```
 }
 ],
 "anomalies": []
 }
}
```

Forensics on API Key

The API Key forensics report includes all activity associated with the specified API Key over a time period. Report information includes a detailed activity trail of accessed URLs, methods, and attacks.

```
{
    "company": "ping identity",
    "name": "api_abs_api_key",
    "description": "This report contains a summary and detailed information on metrics, attacks and anomalies for the
specified api key across all APIs.",
    "earlier_date": "Sat Jan 12 13:30:00:000 2019",
    "later_date": "Tue Dec 31 18:00:00:000 2019",
    "summary": {
        "total_requests": 2621,
        "total_attacks": 1,
        "total_anomalies": 1
    },
    "details": {
        "metrics": {
            "api_key": "finite_api_key",
            "total_requests": 2621,
            "ip_list": [
                {
                    "ip": "192.168.2.2",
                    "total_requests": 457,
                    "devices": {
                        "UNKNOWN": 457
                    },
                    "methods": {
                        "GET": 457
                    },
                    "urls": {
                        "/atm_app/getzipcode": 457
                    },
                    "apis": {
                        "atm_app": 457
                    }
                },
       "attack_types": {
            "Stolen API Key Attack- Per API Key": [
                "all"
            ]
        },
        "anomaly_types": {
            "Stolen API Key Attack- Per API Key": [
                "all"
            ]
        }
    }
}
```


Copyright © 2025 Ping Identity Corporation

Username Forensics

The Username forensics report includes all activity associated with the specified username over a time period. Report information includes a detailed activity trail of accessed URLs, methods, and attacks.


```
{
    "company": "ping identity",
    "name": "api_abs_username",
    "description": "This report contains a summary and detailed information on metrics, attacks and anomalies for the
specified user name across all APIs.",
    "earlier_date": "Sat Jan 12 13:30:00:000 2019",
    "later_date": "Tue Dec 31 18:00:00:000 2019",
    "summary": {
        "total_requests": 109965,
        "total_attacks": 0,
        "total_anomalies": 0
    },
    "details": {
        "metrics": {
            "username": "t4",
            "tokens": [
                "t4MFBkEe",
                "t4GpEkUS",
                "t4ZxUOjb",
                "t4QEvJKT"
            ],
            "total_requests": 109965,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 54983,
                    "devices": {
                        "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110
Safari/537.36": 54983
                    },
                    "methods": {
                        "POST": 54983
                    },
                    "urls": {
                        "/atm_app_oauth": 54983
                    },
                    "apis": {
                        "atm_app_oauth": 54983
                    }
                }
            ]
        },
        "attack_types": {},
        "anomaly_types": {}
    }
}
```


API metrics reporting

The application programming interface (API) metrics report provides information on client request/response activity to the requested API.

Copyright © 2025 Ping Identity Corporation

It includes a summary report and detailed reporting including API access by method.

> **Note**
>
> If API Security Enforcer (ASE) is deployed in sideband mode, then server field in the output shows the Internet Protocol (IP) address as `0.0.0.0`. For ASE deployed in inline mode, the server field shows the IP address of the backend API server.

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_metrics",
 "description": "This report contains metrics for request/response traffic for the specified API",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "api_name": "atm_app_public",
 "req_resp_summary": {
 "api_url": "/atm_app_public",
 "total_requests": 2508,
 "success": 2246,
 "sessions": 2,
 "no_sessions": 1,
 "most_popular_method": "POST",
 "most_popular_device": "UNKNOWN",
 "most_popular_ips": [
 "127.0.0.1",
 "3.1.1.4"
 ],
 "servers": [
 {
 "server": "127.0.0.1:3000",
 "count": 2507
 }
 ]
 },
 "req_resp_details": {
 "api_url": "/atm_app_public",
 "session_details": [
 {
 "session_id": "session_protocol",
 "total_requests": 1,
 "source_ip": [
 {
 "ip": "127.0.0.1",
 "count": 1,
 "method": [
 "GET"
 ]
 }
 ],
 "user_agent": [
 {
 "user_agent": "DOWNLOAD",
 "count": 1
 }
 ],
 "path_info": [
 {
 "path": "/atm_app_public/get400",
 "count": 1
 }
 ],
 "device": [
 {
 "device": "UNKNOWN",
 "count": 1
 }
 ],
 "server": []
```


Copyright © 2025 Ping Identity Corporation

```
},
{
"session_id": "session11",
"total_requests": 2506,
"source_ip": [
{
"ip": "127.0.0.1",
"count": 2506,
"method": [
"DELETE",
"POST",
"PUT",
"GET"
]
}
],
"user_agent": [
{
"user_agent": "DOWNLOAD",
"count": 2506
}
],
"path_info": [
{
"path": "/atm_app_public/post400",
"count": 218
},
{
"path": "/atm_app_public/put400",
"count": 18
},
{
"path": "/atm_app_public/delete200",
"count": 208
},
{
"path": "/atm_app_public/get400",
"count": 14
},
{
"path": "/atm_app_public/put200",
"count": 152
},
{
"path": "/atm_app_public/delete400",
"count": 10
},
{
"path": "/atm_app_public/get200",
"count": 104
},
{
"path": "/atm_app_public/post200",
"count": 1782
}
],
"device": [
{
"device": "UNKNOWN",
"count": 2506
}
],
```


Copyright © 2025 Ping Identity Corporation

```
  "server": [
  {
  "server": "127.0.0.1:3000",
  "count": 2506
  }
  ]
  }
  ],
  "no_session": {
  "request_details": [
  {
  "total_requests": 1,
  "source_ip": [
  {
  "ip": "3.1.1.4",
  "count": 1,
  "method": [
  "GET"
  ]
  }
  ],
  "user_agent": [
  {
  "user_agent": "DOWNLOAD",
  "count": 1
  }
  ],
  "path": "/atm_app_public/get400",
  "device": [
  {
  "device": "UNKNOWN",
  "count": 1
  }
  ],
  "server": [
  {
  "server": "127.0.0.1:3000",
  "count": 1
  }
  ]
  }
  ]
  }
  }
}
```


For more information on ASE sideband mode, see the ASE Admin Guide.

Username based metrics

The username metrics report provides a summary with the total number of usernames, number of requests, tokens and Internet Protocol (IP) address associated with the username.

All the tokens used by the username along with the number of requests for each token is detailed.

Copyright © 2025 Ping Identity Corporation

```
{
  "company": "ping identity",
  "name": "username_metrics",
  "description": "This report contains a summary and detailed username metrics across all APIs",
  "earlier_date": "Tue Oct 08 06:00:00:000 2019",
  "later_date": "Tue Oct 08 06:10:00:000 2019",
  "summary": {
    "usernames": 36697,
    "total_requests": 398776
  },
  "details": [
    {
      "username": "93YgxYHg7B2a9967aZCVRHfc9GEdBBS79tXNWEym",
 "token_list": [
  {
  “token” :  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6IjAwMDEiLCJpc3MiOiJC”,
 “total_requests” : 4
  },
 “token” :  "iZ4Eev2Tutah2pou8uev4kohyiesexai0rool5les8Eilae4aejair”,
 “total_requests” : 2
  }

 ]
“total_requests": 6,
      "ip_list": [
        {
          "ip": "2.63.6.57",
          "total_requests": 6,
          "devices": {
            "UNKNOWN": 6
          },
          "methods": {
            "GET": 6
          },
          "urls": {
            "/accounts/statement": 6
          },
          "apis": {
            "app16": 6
          }
        }
      ]
    }
]
}
```


API Key based metrics

API Behavioral Security (ABS) provides application programming interface (API) key metrics including the total number of API keys and requests across all API keys.

The report also lists the Internet Protocol (IP) address, requesting device information, methods used, Uniform Resource Locator (URL) accessed, and API affected. API key based metrics reporting spans all APIs.

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_key_metrics",
    "description": "This report contains a summary and detailed api key metrics across all APIs",
    "earlier_date": "Mon May 27 13:00:00:000 2019",
    "later_date": "Sun Jun 30 18:00:00:000 2019",
    "summary": {
        "api_keys": 2,
        "total_requests": 3828
    },
    "details": [
        {
            "api_key": "game_api_key",
            "total_requests": 6,
            "ip_list": [
                {
                    "ip": "192.168.2.148",
                    "total_requests": 2,
                    "devices": {
                        "UNKNOWN": 2
                    },
                    "methods": {
                        "GET": 2
                    },
                    "urls": {
                        "/atm_app/getzipcode": 2
                    },
                    "apis": {
                        "atm_app": 2
                    }
                },
                {
                    "ip": "192.168.2.149",
                    "total_requests": 2,
                    "devices": {
                        "UNKNOWN": 2
                    },
                    "methods": {
                        "GET": 2
                    },
                    "urls": {
                        "/atm_app/getzipcode": 2
                    },
                    "apis": {
                        "atm_app": 2
                    }
                },
                {
                    "ip": "192.168.2.146",
                    "total_requests": 2,
                    "devices": {
                        "UNKNOWN": 2
                    },
                    "methods": {
                        "GET": 2
                    },
                    "urls": {
                        "/atm_app/getzipcode": 2
                    },
                    "apis": {
```


Copyright © 2025 Ping Identity Corporation

```
                        "atm_app": 2
                    }
                }
            ]
        },
        {
            "api_key": "uber_api_key",
            "total_requests": 3822,
            "ip_list": [
                {
                    "ip": "192.168.2.2",
                    "total_requests": 457,
                    "devices": {
                        "UNKNOWN": 457
                    },
                    "methods": {
                        "GET": 457
                    },
                    "urls": {
                        "/atm_app/getzipcode": 457
                    },
                    "apis": {
                        "atm_app": 457
                    }
                },
                {
                    "ip": "192.168.2.1",
                    "total_requests": 561,
                    "devices": {
                        "UNKNOWN": 561
                    },
                    "methods": {
                        "GET": 561
                    },
                    "urls": {
                        "/atm_app/getzipcode": 561
                    },
                    "apis": {
                        "atm_app": 561
                    }
                },
                {
                    "ip": "192.168.2.3",
                    "total_requests": 404,
                    "devices": {
                        "UNKNOWN": 404
                    },
                    "methods": {
                        "GET": 404
                    },
                    "urls": {
                        "/atm_app/getzipcode": 404
                    },
                    "apis": {
                        "atm_app": 404
                    }
                },
                {
                    "ip": "192.168.2.5",
                    "total_requests": 2400,
                    "devices": {
                        "UNKNOWN": 2400
```


Copyright © 2025 Ping Identity Corporation

```
                    },
                    "methods": {
                        "GET": 2400
                    },
                    "urls": {
                        "/atm_app/getzipcode": 2400
                    },
                    "apis": {
                        "atm_app": 2400
                    }
                }
            ]
        }
    ]
}
```

OAuth token based metrics

The OAuth2 token metrics report provides a summary with the total number of tokens and requests.

For each token, detailed information on all activity is provided for the time period.

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "oauth_token_metrics",
 "description": "This report contains a summary and detailed oauth token
  metrics across all APIs",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "summary": {
 "tokens": 30,
 "total_requests": 163250
 },
 "details": [
 {
 "token": "token_highresptime",
 "total_requests": 2,
 "ip_list": [
 {
 "ip": "127.0.0.1",
 "total_requests": 2,
 "devices": {
 "UNKNOWN": 2
 },
 "methods": {
 "GET": 2
 },
 "urls": {
 "/2_atm_app_oauth/longresponse": 1,
 "/atm_app_oauth/longresponse": 1
 },
 "apis": {
 "atm_app_oauth": 1,
 "2_atm_app_oauth": 1
 }
 }
 ]
 },
 {
 "token": "token13",
 "total_requests": 7452,
 "ip_list": [
 {
 "ip": "127.0.0.1",
 "total_requests": 7452,
 "devices": {
 "UNKNOWN": 7452
 },
 "methods": {
 "DELETE": 564,
 "POST": 352,
 "GET": 4000,
 "PUT": 2536
 },
 "urls": {
 "/2_atm_app_oauth/put200": 1248,
 "/atm_app_oauth/delete200": 246,
 "/2_atm_app_oauth/put400": 20,
 "/2_atm_app_oauth/get400": 118,
 "/2_atm_app_oauth/get200": 1882,
 "/2_atm_app_oauth/post200": 162,
 "/2_atm_app_oauth/delete200": 246,
```


Copyright © 2025 Ping Identity Corporation

```
"/2_atm_app_oauth/delete400": 36,
"/atm_app_oauth/get200": 1882,
"/atm_app_oauth/post400": 14,
"/2_atm_app_oauth/post400": 14,
"/atm_app_oauth/post200": 162,
"/atm_app_oauth/put400": 20,
"/atm_app_oauth/get400": 118,
"/atm_app_oauth/put200": 1248,
"/atm_app_oauth/delete400": 36
},
"apis": {
"atm_app_oauth": 3726,
"2_atm_app_oauth": 3726
}
}
]
},
{
"token": "token_probing",
"total_requests": 64,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 64,
"devices": {
"UNKNOWN": 64
},
"methods": {
"GET": 64
},
"urls": {
"/2_atm_app_oauth/get400": 32,
"/atm_app_oauth/get400": 32
},
"apis": {
"atm_app_oauth": 32,
"2_atm_app_oauth": 32
}
}
]
},
{
"token": "token_type1memory",
"total_requests": 2,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 2,
"devices": {
"UNKNOWN": 2
},
"methods": {
"PUT": 2
},
"urls": {
"/2_atm_app_oauth/put200": 1,
"/atm_app_oauth/put200": 1
},
"apis": {
"atm_app_oauth": 1,
"2_atm_app_oauth": 1
}
```


Copyright © 2025 Ping Identity Corporation

```
}
]
},
{
"token": "token_contenttype",
"total_requests": 2,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 2,
"devices": {
"UNKNOWN": 2
},
"methods": {
"PUT": 2
},
"urls": {
"/2_atm_app_oauth/put400": 1,
"/atm_app_oauth/put400": 1
},
"apis": {
"atm_app_oauth": 1,
"2_atm_app_oauth": 1
}
}
]
},
{
"token": "token_method",
"total_requests": 2,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 2,
"devices": {
"UNKNOWN": 2
},
"methods": {
"HEAD": 2
},
"urls": {
"/2_atm_app_oauth/get400": 1,
"/atm_app_oauth/get400": 1
},
"apis": {
"atm_app_oauth": 1,
"2_atm_app_oauth": 1
}
}
]
}
]
}
```


List valid URL

The List Valid Uniform Resource Locator (URL) report includes all URLs, access count, and allowed methods for a specified application programming interface (API).

Copyright © 2025 Ping Identity Corporation

The report provides insight into the activity on each API URL.


```
{
 "company": "ping identity",
 "name": "api_url_list",
 "description": "This report contains list of valid URL for the specified API",
 "api_name": "shop",
 "host_name": "app",
 "api_url": "shopapi",
 "allowed_methods": [
 "GET",
 "PUT",
 "POST",
 "DELETE",
 "HEAD"
 ],
 "url_list": [
 {
 "protocol": "HTTP/1.1",
 "urls": [
 {
 "url": "/shopapi/post",
 "total_count": 2009,
 "methods": [
 {
 "method": "POST",
 "count": 2009
 }
 ]
 },
 {
 "url": "/shopapi/login",
 "total_count": 2956,
 "methods": [
 {
 "method": "POST",
 "count": 2956
 }
 ]
 },
 {
 "url": "/shopapi/login?username=v1&amp;password=v2",
 "total_count": 87,
 "methods": [
 {
 "method": "POST",
 "count": 87
 }
 ]
 },
 {
 "url": "/shopapi/put",
 "total_count": 2159,
 "methods": [
 {
 "method": "PUT",
 "count": 2159
 }
```


Copyright © 2025 Ping Identity Corporation

Hacker's URL

The List Invalid Uniform Resource Locator (URL) or hacker's URL report provide information on the four types of invalid URLs.

The four invalid URLs are: irregular URLs, system commands, buffer overflow, and SQL injection.

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_cookie",
 "description": "This report contains a summary and detailed information on metrics,
  attacks and anomalies for the specified cookie across all APIs.",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "summary": {
 "total_requests": 32768,
 "total_attacks": 3,
 "total_anomalies": 1
 },
 "details": {
 "metrics": [
 {
 "session_id": "session_extremeactivity",
 "start_time": "Thu Feb 15 14:04:46:001 2018",
 "end_time": "Thu Feb 15 14:05:02:994 2018",
 "total_requests": 16384,
 "source_ip": [
 {
 "ip": "127.0.0.1",
 "count": 16384,
 "method": [
 "GET"
 ]
 }
 ],
 "user_agent": [
 {
 "user_agent": "DOWNLOAD",
 "count": 16384
 }
 ],
 "path_info": [
 {
 "path": "/atm_app_public/get200",
 "count": 16384
 }
 ],
 "device": [
 {
 "device": "UNKNOWN",
 "count": 16384
 }
 ],
 "server": [
 {
 "server": "127.0.0.1:3000",
 "count": 16384
 }
 ]
 },
 {
 "session_id": "session_extremeactivity",
 "start_time": "Thu Feb 15 14:13:45:795 2018",
 "end_time": "Thu Feb 15 14:14:35:268 2018",
 "total_requests": 16384,
 "source_ip": [
 {
```


Copyright © 2025 Ping Identity Corporation

```
"ip": "127.0.0.1",
"count": 16384,
"method": [
"GET"
]
}
],
"user_agent": [
{
"user_agent": "DOWNLOAD",
"count": 16384
}
],
"path_info": [
{
"path": "/2_atm_app_public/get200",
"count": 16384
}
],
"device": [
{
"device": "UNKNOWN",
"count": 16384
}
],
"server": [
{
"server": "127.0.0.1:3000",
"count": 16384
}
]
}
],
"attack_types": {
"Data Exfiltration Attack": [
"2_atm_app_public",
"atm_app_public"
],
"Extreme Client Activity Attack": [
"2_atm_app_public",
"atm_app_public"
],
"Extreme App Activity Attack": [
"2_atm_app_public",
"atm_app_public"
]
},
"anomaly_types": {
"Stolen Cookie Anomaly": [
"2_atm_app_public",
"atm_app_public"
]
}
}
}
```


Copyright © 2025 Ping Identity Corporation

Backend error reporting

The Backend Error Response Codes report provides information for each error code including client Internet Protocol (IP), server IP, and requested Uniform Resource Locator (URL).

API Behavioral Security (ABS) reports on a per API basis for the following error codes:

- 403: Forbidden
- 404: Not Found
- 500: Internal Server Error
- 503: Service Unavailable
- 504: Gateway Timeout


```
{
 "company": "ping identity",
 "name": "api_backend_errors",
 "description": "This report contains details of backend error codes for
  the specified API",
 "later_date": "Sun Feb 05 13:20:00:000 2017",
 "earlier_date": "Wed Feb 01 08:20:00:000 2017",
 "api_name": "atmapp",
 "backend_error_summary": [
 {
 "error_code": "403",
 "error": "Forbidden",
 "count": 0
 },
 {
 "error_code": "404",
 "error": "Not Found",
 "count": 0
 },
truncated
 ],
 "backend_error_details": [
 {
 "error_code": "500",
 "details": [
 {
 "server": "192.168.11.164:3001",
 "request_url": "/atmapp/zipcode",
 "request_ip": "100.64.5.183:24078",
 "request_cookie": ""
 },
 {
 "server": "192.168.11.164:3003",
 "request_url": "/atmapp/zipcode",
 "request_ip": "100.64.19.136:61494",
 "request_cookie": "JSESSIONID=5GMNKOGNGP6FCKF9"
 },
```


Copyright © 2025 Ping Identity Corporation

API DoS and DDoS threshold

API DoS and DDoS threshold 11.

API Flow Control reports on API Security Enforcer (ASE) configured flow control thresholds that are exceeded. The reporting is done on the following parameters:

- Client Spike – inbound client traffic rate
- Server Spike – aggregate traffic to an application programming interface (API) service
- Connection Queued – connection requests queued due to server at concurrent connection limit
- Bytes-in Spike – WebSocket aggregate inbound traffic exceeds limit
- Bytes-out Spike - WebSocket aggregate outbound traffic exceeds limit

> **Note**
>
> API DoS and DDoS threshold and reporting is only available when ASE is deployed in inline mode.

For a specified API, the flow control API provides a summary of thresholds exceeded and detailed reporting on each flow control threshold exceeded:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_flowcontrol",
 "description": "This report contains flow control information for the specified API",
 "earlier_date": "Thu Jan 25 18:00:00:000 2018",
 "later_date": "Fri Dec 28 18:00:00:000 2018",
 "api_name": "atm_app_private",
 "server_spike_ip_count": 0,
 "summary": {
 "client_spike": 990,
 "server_spike": 0,
 "connection_queued": 0,
 "connection_quota_exceeded": 0
 },
 "details": {
 "client_spike": [
 {
 "request_time": "Mon Jan 29 13:43:20:227 2018",
 "connection_id": "2081496566",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:228 2018",
 "connection_id": "1902346354",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:228 2018",
 "connection_id": "1999376747",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:228 2018",
 "connection_id": "2009947644",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:228 2018",
 "connection_id": "934081844",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:227 2018",
 "connection_id": "2081496566",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:228 2018",
 "connection_id": "1902346354",
 "source_ip": "3.1.1.2",
 "destination_api": "/atm_app_private/get400"
 },
 {
 "request_time": "Mon Jan 29 13:43:20:228 2018",
```


Copyright © 2025 Ping Identity Corporation

```
  "connection_id": "1999376747",
  "source_ip": "3.1.1.2",
  "destination_api": "/atm_app_private/get400"
  },
  {
  "request_time": "Mon Jan 29 13:43:20:228 2018",
  "connection_id": "2009947644",
  "source_ip": "3.1.1.2",
  "destination_api": "/atm_app_private/get400"
  },
  {
  "request_time": "Mon Jan 29 13:43:20:228 2018",
  "connection_id": "934081844",
  "source_ip": "3.1.1.2",
  "destination_api": "/atm_app_private/get400"
  }
  ],
  "server_spike": [],
  "connections_queued": [],
  "connection_quota_exceeded": []
  }
 }
```


API reports using Postman

Multiple options are available for accessing the API Behavioral Security (ABS) REST application programming interface (API) reporting.

These options include the following:

- Postman App
- Java, Python, C Sharp, or similar languages.
- Java client program (such as Jersey)
- C sharp client program (such as RestSharp)

For the Postman application, Ping Identity provides configuration files which are used by Postman to access the ABS REST API JavaScript Object Notation (JSON) information reports. Make sure to install Postman 6.2.5 or higher.

Using an ABS self-signed certificate with Postman

API Behavioral Security (ABS) ships with a self-signed certificate. To use Postman with the ABS self-signed certificate, disable certificate verification in Postman.

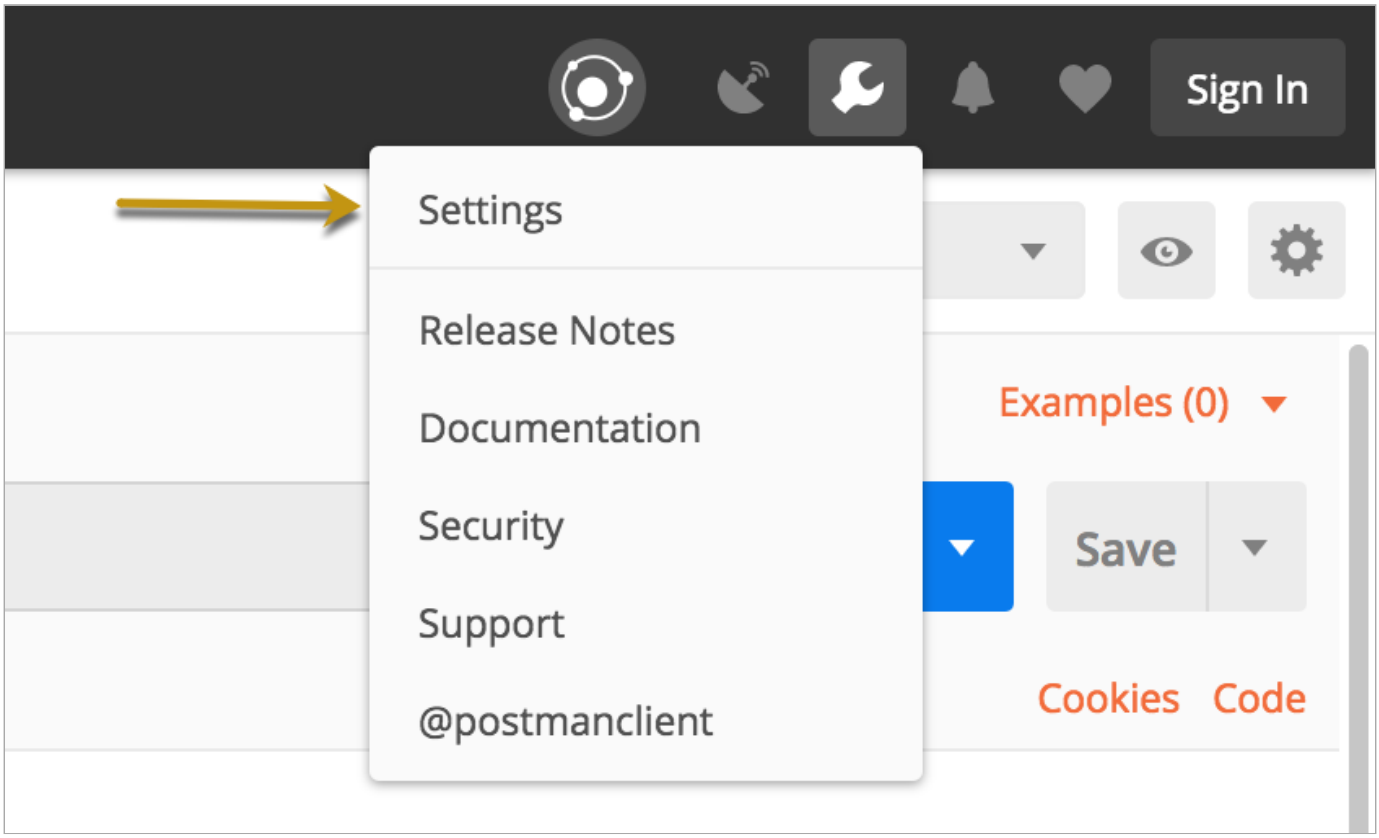
About this task

To disable certificate verification:

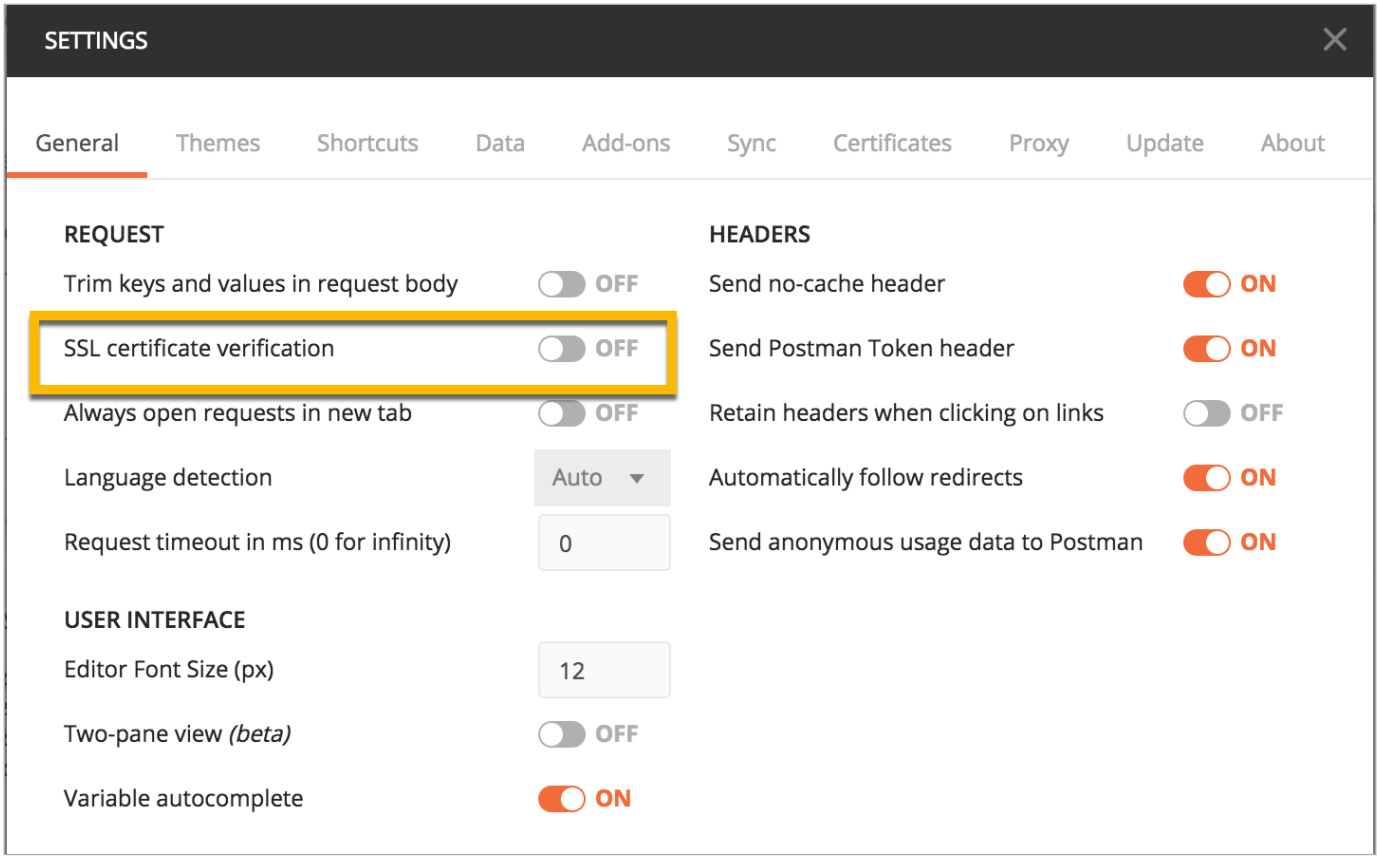
Steps

1. Click the Wrench icon on the top-right corner of the Postman client.
2. In the list, select Settings.

Copyright © 2025 Ping Identity Corporation



3. In theSettings window, click theSSL certificate verification toggle to disable SSL certificate verification.



Copyright © 2025 Ping Identity Corporation

Viewing ABS reports in Postman

You can view API Behavioral Security (ABS) reports in Postman.

About this task

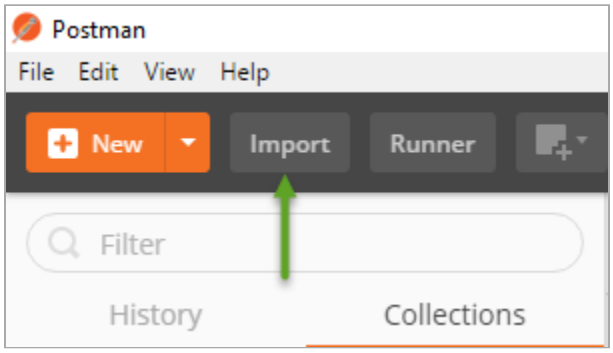
To view the reports:

Steps

1. Download the `ABS_5.0_Environment` and `ABS_5.0_Reports` JSON files from API Reports Using Postman folder on the Ping Identity Download site.

> **Note**
>
> These configuration files will be used by Postman.

2. Download and install the Postman application 6.2.5 or later.
3. In Postman, click Import and select the two Ping Identity files that you downloaded in step 1.



4. After importing the files, click the Gear icon in the upper right corner.
5. In the Manage Environments window, click ABS_5.0_Environment.
6. Configure the following values and then click Update.

Option	Value
Server	IP address of the ABS node for which the `dashboard_node` was set to `true` in the `abs.properties` file.
Port	Port number of the ABS node.
Access_Key_Headerand Secret_Key_Header	Use the admin user or restricted user header. A restricted user sees obfuscated value of OAuth token, cookie, and API keys. For more information of different types of users, see ABS users for API reports.

Copyright © 2025 Ping Identity Corporation

Option	Value
Access_Key and Secret_Key	The access key and secret key configured in the `opt/pingidentity/mongo/abs_init.js` file for either admin or restricted user. **Note** Make sure that the access key and secret key corresponds to the admin or restricted user header configured.
API_Name	The name of the API for which you want to generate the reports.
Later_Date	A date that is more recent. For example, if the query range is between March 12 and March 14, then the later date would be March 14.
Earlier_Date	A date that is earlier. For example, if the query range is between March 12 and March 14, then the earlier date would be March 12. **Note** Do not edit any fields that start with the word `System`.

7. In the main Postman window, select the report to display on the left column and then click Send.

   For detailed information on each API call and the JSON report response, see the ABS external REST APIs section.

ABS CLI commands

The API Behavioral Security (ABS) command-line interface (CLI) provides basic commands and obfuscation commands.

Basic commands

Command	Definition	Syntax
Start ABS	Starts ABS. Run the command from `/opt/pingidentity/abs/bin` directory.	`./start.sh`
Stop ABS	Stops ABS. Run the command from `/opt/pingidentity/abs/bin` directory.	`./stop.sh`
Help	Displays `cli.sh` help.	`./cli.sh help`

Copyright © 2025 Ping Identity Corporation

Command	Definition	Syntax
Update Password	Changes ABS admin password.	`./cli.sh update_password \{-u admin}`
Update Keys	Updates access and secret keys.	`./cli.sh -u admin -p admin update_keys -ak <access key> -sk < secret key>` **Note** You should use the **`update_keys`** command to change the keys. However, you can directly edit the `abs_init.js` file when changing the default access and secret keys for the first time after installing ABS AI engine.

Obfuscation commands

Command	Definition	Syntax
Generate Master Key	Generates the master obfuscation key `abs_master.key`.	`./cli.sh -u admin -p admin generate_obfkey`
Obfuscate Password	Obfuscates the passwords configured in various configuration files.	`./cli.sh -u admin -p admin obfuscate_keys`

ABS REST API format

API Behavioral Security (ABS) provides external REST APIs.

External REST APIs are used to access JavaScript Object Notation (JSON) reports providing deep insight into the following:

- Attack Forensics and Compliance Reporting – attacks and anomalous behavior on APIs.
- API Metrics – API client and traffic details.
- Administrative – ABS system information.
- API Security Enforcer – decoy API, blocked connections, flow control, and backend error reporting.

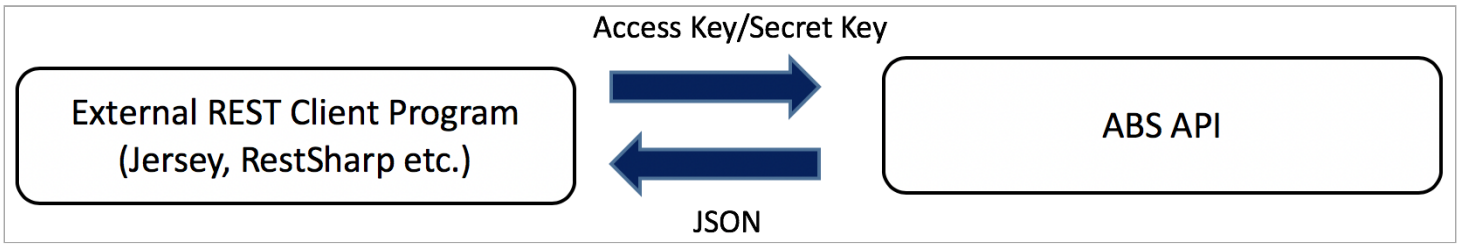
A REST client can securely query each ABS API and receive data back in JSON format. REST client program options include using:

- Postman App for Google Chrome browser.
- Java, Python, C Sharp, or similar languages.
- Java client program (for example, Jersey).

Copyright © 2025 Ping Identity Corporation

- C sharp client program (for example, RestSharp).

The diagram shows the process for a REST API client to connect to an ABS API.



ABS API query format

ABS API offers a common format with a consistent syntax for request parameters. Detailed information and format of all ABS REST APIs are included in ABS external REST APIs.

Query parameters for most APIs are shown in the table below:

Field	Description
`api_name`	The API name to query for results.
`earlier_date`	The time to check for results going back in time. For example, to check results from April, 10, 6:00 p.m. to April, 14, 3:00 p.m., the `earlier_date` would be April, 10, 6:00 p.m.
`later_date`	The time to check the results back in time. For example, to check results from April 10 , 6:00 p.m. to April, 14, 3:00 p.m., the `later_date` would be April, 14, 6:00 p.m.

The following `access_key` and `secret_key` are the keys that were defined in the `abs_init.js` file.

> **Note**
>
> The ":" (colon) is a restricted character and cannot be used in access and secret key.

- `x-abs-ak` and `x-abs-ak-ru:` access_key
- `x-abs-sk` and `x-abs-sk-ru:` secret_key

> **Note**
>
> The start and end time are based on the log file data, that is, the local time where data was captured and not of the location where results are analyzed.

ABS external REST APIs

The following is a list of Ping Identity API Behavioral Security (ABS) application programming interface (API).

Copyright © 2025 Ping Identity Corporation

The sample outputs produced are for the Admin user. You can generate the output for the restricted user as well where the cookie, token, and API keys are obfuscated. For more information on different type of users for the ABS External REST APIs, see ABS Users for API Reports and Dashboard**.**

> **Note**
>
> The ":" (colon) is a restricted character and cannot be used in access and secret key headers in ABS external REST APIs.

- Admin REST API
- TTL Update REST API
- Global Config Update REST API
- Discovery REST API
- Threshold REST API
- Reset Trained API
- Decoy API
- IP Metrics REST API
- API Key Metrics REST API
- Token Metrics REST API
- Username Metrics REST API
- Anomalies REST API
- Token Forensics REST API
- IP Forensics REST API
- Cookie Forensics REST API
- API Key Forensics API REST
- Username Forensics REST API
- Attack Type REST API
- Flow Control REST API
- Blocked Connection REST API
- Backend Error REST API
- List Valid URLs REST API
- List Hacker's URLs REST API

Copyright © 2025 Ping Identity Corporation

Admin REST API

Admin application programming interface (API) is used to fetch the list of nodes in the API Behavioral Security (ABS) cluster, Mongo DB Nodes, the status of each node (CPU, memory, file System etc) and logs processed that are sent by all API Security Enforcer nodes.

`URL: /v5/abs/admin`

`Method: GET`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample curl command:

```
curl --location --request GET 'https://<IP Address>:8080/v5/abs/admin' \
--header 'x-abs-ak: abs_ak' \
--header 'x-abs-sk: abs_sk'
```

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, and ASE logs",
    "license_info": {
        "tier": "Subscription",
        "expiry": "Mon Jan 01 00:00:00 UTC 2024",
        "max_transactions_per_month": 1000000000,
        "current_month_transactions": 0,
        "max_transactions_exceeded": false,
        "expired": false
    },
    "across_api_prediction_mode": false,
    "poc": true,
    "api_discovery": {
        "status": false
    },
    "apis": [
        {
            "api_name": "rest_api_decoy_outcontext",
            "host_name": "",
            "url": "/decoy",
            "api_type": "decoy-outcontext",
            "creation_date": "Mon May 31 03:41:10 UTC 2021",
            "servers": 0,
            "protocol": "http",
            "cookie": "",
            "token": false,
            "training_started_at": "n/a",
            "training_duration": "n/a",
            "prediction_mode": false,
            "apikey_header": "",
            "apikey_qs": "",
            "jwt": {
                "location": "",
                "username": "",
                "clientid": ""
            },
            "username_header": ""
        },
        {
            "api_name": "rest_api",
            "host_name": "",
            "url": "/rest",
            "api_type": "regular",
            "creation_date": "Mon May 31 03:41:10 UTC 2021",
            "servers": 2,
            "protocol": "http",
            "cookie": "",
            "token": false,
            "training_started_at": "Mon May 31 03:42:46 UTC 2021",
            "training_duration": "20 hours",
            "prediction_mode": false,
            "apikey_header": "",
            "apikey_qs": "",
            "jwt": {
                "location": "",
                "username": "",
                "clientid": ""
            },
```


Copyright © 2025 Ping Identity Corporation

```
            "username_header": ""
        },
        {
            "api_name": "rest_api_decoy_incontext",
            "host_name": "",
            "url": "/restdecoy",
            "api_type": "decoy-incontext",
            "creation_date": "Mon May 31 03:41:10 UTC 2021",
            "servers": 2,
            "protocol": "http",
            "cookie": "",
            "token": false,
            "training_started_at": "Mon May 31 03:42:46 UTC 2021",
            "training_duration": "20 hours",
            "prediction_mode": false,
            "apikey_header": "",
            "apikey_qs": "",
            "jwt": {
                "location": "",
                "username": "",
                "clientid": ""
            },
            "username_header": ""
        },
        {
            "api_name": "root_api",
            "host_name": "",
            "url": "/",
            "api_type": "regular",
            "creation_date": "Mon May 31 03:41:10 UTC 2021",
            "servers": 1,
            "protocol": "http",
            "cookie": "JSESSIONID",
            "token": false,
            "training_started_at": "n/a",
            "training_duration": "n/a",
            "prediction_mode": false,
            "apikey_header": "X-API-KEY",
            "apikey_qs": "",
            "jwt": {
                "location": "h:authorization:bearer",
                "username": "username",
                "clientid": "client_id"
            },
            "username_header": ""
        }
    ],
    "abs_cluster": {
        "abs_nodes": [
            {
                "node_id": "937e8553-ccc6-419b-b24d-c4f9a5d46632",
                "os": "Red Hat Enterprise Linux Server - ",
                "cpu": "8",
                "memory": "15G",
                "filesystem": "6%",
                "timezone": "UTC",
                "bootup_date": "Fri May 21 04:57:08:502 UTC 2021"
            },
            {
                "node_id": "d224a4f9-3a17-423a-a7b3-571c8143cfff",
                "os": "Red Hat Enterprise Linux Server - VMware, Inc.",
                "cpu": "16",
```


Copyright © 2025 Ping Identity Corporation

```
                "memory": "62G",
                "filesystem": "2%",
                "timezone": "UTC",
                "bootup_date": "Mon May 31 03:39:14:818 UTC 2021"
            }
        ],
        "mongodb_nodes": [
            {
                "node_ip": "172.16.40.165:27017",
                "status": "primary"
            }
        ]
    },
    "ase_logs": [
        {
            "ase_node": "53de980b-df8e-467a-9328-04e33497d2cd",
            "last_connected": "Mon May 31 03:41:15 UTC 2021",
            "logs": {
                "start_time": "Mon May 31 03:41:15 UTC 2021",
                "end_time": "Mon May 31 03:41:15 UTC 2021",
                "gzip_size": "13.96MB"
            }
        }
    ],
    "percentage_diskusage_limit": "80%",
    "scale_config": {
        "scale_up": {
            "cpu_threshold": "70%",
            "cpu_monitor_interval": "30 minutes",
            "memory_threshold": "70%",
            "memory_monitor_interval": "30 minutes",
            "disk_threshold": "70%",
            "disk_monitor_interval": "30 minutes"
        },
        "scale_down": {
            "cpu_threshold": "10%",
            "cpu_monitor_interval": "30 minutes",
            "memory_threshold": "10%",
            "memory_monitor_interval": "30 minutes",
            "disk_threshold": "10%",
            "disk_monitor_interval": "30 minutes"
        }
    },
    "attack_ttl": {
        "ids": [
            {
                "id": "ip",
                "ttl": 0
            },
            {
                "id": "cookie",
                "ttl": 0
            },
            {
                "id": "access_token",
                "ttl": 0
            },
            {
                "id": "api_key",
                "ttl": 0
            },
            {
```


Copyright © 2025 Ping Identity Corporation

```
                "id": "username",
                "ttl": 0
            }
        ]
    }
}
```

Discovery REST API

The Discovery application programming interface (API) discovers all the APIs that are available in your API ecosystem.

`Method: GET`

`URL: /v4/abs/discovery`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_discovery_summary",
    "description": "This report contains summary of discovered APIs",
    "summary": [
        {
            "api_name": "api_0",
            "host": "bothcookientoken.com",
            "basePath": "/path1",
            "created": "Fri Mar 06 09:29:51:591 2020",
            "updated": "Fri Mar 06 09:50:03:372 2020"
        },
        {
            "api_name": "api_1",
            "host": "path5",
            "basePath": "/path1/path2/path3",
            "created": "Fri Mar 06 10:59:38:975 2020",
            "updated": "Fri Mar 06 11:36:45:596 2020"
        },
        {
            "api_name": "api_10",
            "host": "pathx",
            "basePath": "/path1/path2/path3",
            "created": "Fri Mar 06 10:59:57:320 2020",
            "updated": "Fri Mar 06 13:19:24:680 2020"
        },
        {
            "api_name": "api_11",
            "host": "path8",
            "basePath": "/path1",
            "created": "Fri Mar 06 10:59:39:392 2020",
            "updated": "Fri Mar 06 13:19:23:951 2020"
        },
        {
            "api_name": "api_12",
            "host": "path3",
            "basePath": "/path1/path2/path3",
            "created": "Fri Mar 06 10:59:38:672 2020",
            "updated": "Fri Mar 06 13:19:23:152 2020"
        },
        {
            "api_name": "api_13",
            "host": "path4",
            "basePath": "/path1/path2/path3",
            "created": "Fri Mar 06 10:59:38:824 2020",
            "updated": "Fri Mar 06 11:36:45:452 2020"
        },
        {
            "api_name": "api_14",
            "host": "path5",
            "basePath": "/path1/path2/path3/path4/path5",
            "created": "Fri Mar 06 11:59:14:804 2020",
            "updated": "Fri Mar 06 12:18:24:732 2020"
        },
        {
            "api_name": "api_15",
            "host": "pathx",
            "basePath": "/path1/path2/path3/path4",
            "created": "Fri Mar 06 11:59:16:092 2020",
            "updated": "Fri Mar 06 13:19:25:283 2020"
```


Copyright © 2025 Ping Identity Corporation

```
        },
        {
            "api_name": "api_16",
            "host": "pathx",
            "basePath": "/path1/path2/path3/path4/path5",
            "created": "Fri Mar 06 11:59:16:244 2020",
            "updated": "Fri Mar 06 12:18:26:227 2020"
        },
        {
            "api_name": "api_17",
            "host": "path6",
            "basePath": "/path1/path2/path3/path4/path5/path6",
            "created": "Fri Mar 06 11:59:14:952 2020",
            "updated": "Fri Mar 06 12:18:24:876 2020"
        },
        {
            "api_name": "api_18",
            "host": "pathx",
            "basePath": "/path1/path2/path3/path4/path5/path6",
            "created": "Fri Mar 06 11:59:16:396 2020",
            "updated": "Fri Mar 06 12:18:26:532 2020"
        },
        {
            "api_name": "api_19",
            "host": "path7",
            "basePath": "/path1/path2/path3/path4/path5/path6",
            "created": "Fri Mar 06 11:59:15:096 2020",
            "updated": "Fri Mar 06 12:18:25:028 2020"
        },
        {
            "api_name": "api_9",
            "host": "path2",
            "basePath": "/path1/path2",
            "created": "Fri Mar 06 10:59:00:616 2020",
            "updated": "Fri Mar 06 13:19:23:003 2020"
        }
    ]
}
```


Decoy REST API

Decoy application programming interface (API) provides information about the Internet Protocol (IP) address that accessed the decoy Uniform Resource Locator (URL) along with the method used to access the decoy URL.

It also reports about the type of device that was used to access the decoy URL.

`Method: GET`

`URL: /v4/abs/decoy?later_date<>&earlier_date<>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Copyright © 2025 Ping Identity Corporation

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "decoy_api_metrics",
 "description": "This report contains detailed information on client access to each decoy API",
 "earlier_date": "Tue Jan 11 17:50:00:000 2018",
 "later_date": "Tue Jan 11 18:00:00:000 2018",
 "api_name": "atmapp",
 "api_type": "decoy-incontext",
 "decoy_url": [
 "/atmapp/decoy"
 ],
 "summary": [
 {
 "decoy_url": "/atmapp/decoy",
 "unique_ip_count": 122,
 "total_requests": 240,
 "most_used_methods": {
 "GET": 88,
 "DELETE": 32,
 "ABDU": 32,
 "POST": 30,
 "PUT": 26
 },
 "most_used_ips": {
 "100.64.9.37": 4,
 "100.64.10.79": 4,
 "100.64.31.183": 2,
 "100.64.20.213": 2,
 "100.64.34.239": 2
 },
 "most_used_devices": {
 "UBUNTU": 76,
 "MAC_OS_X": 69,
 "WINDOWS_7": 61,
 "WINDOWS_XP": 34
 },
 "most_used_content_types": {
 "UNKNOWN": 184,
 "multipart/form-data": 56
 }
 }
 ],
 "details": [
 {
 "decoy_url": "/atmapp/decoy",
 "source_ip": [
 {
 "ip": "100.64.31.183",
 "total_requests": 2,
 "method_count": {
 "GET": {
 "count": 2
 }
 },
 "url_count": {
 "/atmapp/decoy": 2
 }
 },
 {
 "ip": "100.64.14.28",
```


Copyright © 2025 Ping Identity Corporation

```
"total_requests": 2,
"method_count": {
"POST": {
"count": 2,
"payload_characteristics": {
"multipart/form-data": [
"354 bytes"
]
}
}
},
"url_count": {
"/atmapp/decoy": 2
}
},
{
"ip": "100.64.0.55",
"total_requests": 2,
"method_count": {
"GET": {
"count": 2
}
},
"url_count": {
"/atmapp/decoy": 2
}
},
{
"ip": "100.64.20.152",
"total_requests": 2,
"method_count": {
"DELETE": {
"count": 2
}
},
"url_count": {
"/atmapp/decoy": 2
}
}
]
}
]
}
```


Threshold REST API

API Behavioral Security (ABS) provides Threshold REST application programming interface (API) for checking and updating attack thresholds.

It helps to identify and tune thresholds false positives. For more information see, Tune thresholds for false positives.

The following are the methods of Threshold REST APIs:

- GET Threshold
- PUT Threshold

Copyright © 2025 Ping Identity Corporation

GET Threshold

The GET method in Threshold application programming interface (API) fetches the threshold values for attack types.

`Method: GET`

`URL for an API: /v4/abs/attack/threshold?api=<api_name>`

`URL for across API: /v4/abs/attack/threshold?id=<type_id>` .

The API name is not specified in the URL for fetching the threshold value. Type ID is the attack ID.

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample response for an API:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_threshold",
    "description": "This report contains threshold settings for all the across API Attack IDs",
    "thresholds": [
         {
            "id": 1,
            "type": "data_exfiltration_attack",
            "user": {
                "A": {
                    "tn": "18",
                    "tx": "20"
                },
                "B": {
                    "tn": "18",
                    "tx": "20"
                }
            },
            "system": {
                "A": {
                    "tn": "22",
                    "tx": "24"
                },
                "B": {
                    "tn": "4",
                    "tx": "6"
                },
                "C": {
                    "tn": "2",
                    "tx": "4"
                }
            }
        },
        {
            "id": 2,
            "type": "single_client_login_attack",
            "system": {
                "A": {
                    "tn": "5",
                    "tx": "7"
                },
                "B": {
                    "tn": "5",
                    "tx": "7"
                }
            }
        },
}
```


Sample Response for across API:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_threshold",
    "description": "This report contains threshold settings for the specified API",
    "api_name": "access_token",
    "threshold": [
        {
            "type": "extended_stolen_access_token",
            "system": {
                "A": {
                    "tn": "2",
                    "tx": "na"
                },
                "B": {
                    "tn": "1",
                    "tx": "na"
                },
                "C": {
                    "tn": "1",
                    "tx": "na"
                }
            }
        }
    ]
}
```


PUT Threshold

The PUT method in Threshold application programming interface (API) is used to set the threshold values for attack types.

If you set the mode to `system`, the user set values are dropped. If you move the mode back to `user`, you would need to configure the threshold values again. For more information on manually setting threshold values, see Manually set thresholds.

`Method: PUT`

`URL: /v4/abs/attack/threshold`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Input for an API:

Copyright © 2025 Ping Identity Corporation

```
{
 "api_name" : "atmapp",
 "mode":  "system",
 "ioc_threshold": [
 {
 "type": "api_memory_post",
 "variable": "A",

 },
 {
 "type": "api_memory_put",
 "variable": "B"
 }
 ]
}
```

The following is the response when the threshold values are set:

```
{
    "status_code": "SUCCESS",
    "message": "attack threshold updated"
}
```

Sample Input for across API:

```
{
 "id":"18",
 "mode":  "user",
 "ioc_threshold": [
{
     "type": "extended_probing_replay_cookie",
     "variable": "A",
     "tn": "25",
     "tx": "28"
   },{
     "type": "extended_probing_replay_cookie",
     "variable": "B",
     "tn": "3",
     "tx": "4"
   }
]
}
```

The following is the response when the threshold values are set:

```
{
    "status_code": "SUCCESS",
    "message": "attack threshold updated"
}
```


Copyright © 2025 Ping Identity Corporation

Metrics REST API

The Metrics application programming interface (API) is used to fetch API Traffic metrics.

The response contains request count for each API, bad request count, request success, failure count, and so on.

> **Note**
>
> If API Security Enforcer (ASE) is deployed in sideband mode, then server field in the output shows the Internet Protocol (IP) address as `0.0.0.0`. For ASE deployed in inline mode, the server field shows the IP address of the backend API server. For more information on ASE sideband mode, see the ASE Admin Guide.

`Method: GET`

`URL: /v4/abs/metrics?later_date=<>&earlier_date=<>api=<api_name>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_metrics",
 "description": " This report contains metrics for request/response traffic
  for the specified API",
 "earlier_date": "Mon Jan 13 18:00:00:000 2018",
 "later_date": "Wed Jan 15 18:00:00:000 2018",
 "api_name": "shop",
 "req_resp_summary": {
 "api_url": "shopapi",
 "total_requests": 342102,
 "success": 279360,
 "sessions": 0,
 "no_sessions": 342102,
 "most_popular_method": "GET",
 "most_popular_device": "MAC_OS_X",
 "most_popular_ips": [
 "10.10.1.38",
 "10.10.1.39",
 "10.10.1.37"
 ]
 "servers": [
 {
 "server": "192.168.11.164:3001",
 "count": 5357
 },
 {
 "server": "192.168.11.164:3002",
 "count": 5354
 },
 {
 "server": "192.168.11.164:3003",
 "count": 5358
 },
 {
 "server": "192.168.11.164:3004",
 "count": 1667
 }
 ]
 },
 "req_resp_details": {
 "api_url": "shopapi",
 "session_details": [],
 "no_session": {
 "request_details": [
 {
 "total_requests": 14865,
 "source_ip": [
 {
 "ip": "10.10.1.24",
 "count": 152,
 "method": [
 "POST"
 ]
 },
 {
 "ip": "10.10.1.71",
 "count": 482,
 "method": [
 "PUT"
```


Copyright © 2025 Ping Identity Corporation

```
]
}
],
"user_agent": [
{
"user_agent": "SAFARI",
"count": 7187
},
{
"user_agent": "FIREFOX",
"count": 12536
},
{
"user_agent": "MOZILLA",
"count": 5509
},
{
"user_agent": "CHROME",
"count": 29241
}
],
"server": [
{
"server": "192.168.11.164:3001",
"count": 723
},
{
"server": "192.168.11.164:3002",
"count": 689
},
{
"server": "192.168.11.164:3003",
"count": 749
},
{
"server": "192.168.11.164:3004",
"count": 237
}
]
"path": "/shopapi/put",
"device": [
{
"device": "WINDOWS_8",
"count": 8338
},
{
"device": "MAC_OS_X",
"count": 14276
},
{
"device": "WINDOWS_XP",
"count": 5990
},
{
"device": "UBUNTU",
"count": 6546
}
]
},
{
"total_requests": 2,
"source_ip": [
```


Copyright © 2025 Ping Identity Corporation

```
  {
  "ip": "10.10.1.69",
  "count": 2,
  "method": [
  "GET"
  ]
  }
  ],
  "user_agent": [
  {
  "user_agent": "CHROME",
  "count": 2
  }
  ],
  "path": "/shopapi/get/etc",
  "device": [
  {
  "device": "MAC_OS_X",
  "count": 3
  }
  ]
  }
  ]
  }
  }
 }
```


API Key Metrics REST API

The application programming interface (API) Key-based Metrics API is used to fetch the metrics for API Keys across all APIs.

`Method: GET`

`URL: /v4/abs/apikeys?later_date=<yy-mm-dd>T<hh:mm>&earlier_date==<yy-mm-dd>T<hh:mm>`

	Header	Value
Access Key	`x-abs-ak`	`<string>`
Secret Key	`x-abs-sk`	`<string>`

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_key_metrics",
 "description": "This report contains a summary and detailed api key
  metrics across all APIs",
 "earlier_date": "Fri Jan 19 13:00:00:000 2018",
 "later_date": "Sat Jan 20 18:00:00:000 2018",
 "summary": {
 "api_keys": 325,
 "total_requests": 329
 },
 "details": [
 {
 "api_key": "87FYNG7Q8KP1V03O",
 "total_requests": 1,
 "ip_list": [
 {
 "ip": "100.64.5.79",
 "total_requests": 1,
 "devices": {
 "MAC_OS_X": 1
 },
 "methods": {
 "DELETE": 1
 },
 "urls": {
 "/apikeyheader/zipcode": 1
 },
 "apis": {
 "apikeyheader": 1
 }
 }
 ]
 },
 {
 "api_key": "NW0ODLM68PFQ3XTL",
 "total_requests": 1,
 "ip_list": [
 {
 "ip": "100.64.20.62",
 "total_requests": 1,
 "devices": {
 "WINDOWS_XP": 1
 },
 "methods": {
 "DELETE": 1
 },
 "urls": {
 "/apikeyheader/zipcode": 1
 },
 "apis": {
 "apikeyheader": 1
 }
 }
 ]
 },
 {
 "api_key": "86ELLUSN6RAHEPF7",
 "total_requests": 1,
 "ip_list": [
```


Copyright © 2025 Ping Identity Corporation

```
  {
  "ip": "100.64.17.79",
  "total_requests": 1,
  "devices": {
  "MAC_OS_X": 1
  },
  "methods": {
  "GET": 1
  },
  "urls": {
  "/apikeyheader/zipcode": 1
  },
  "apis": {
  "apikeyheader": 1
  }
  }
  ]
  },
  {
  "api_key": "5JSKZZ53TGBQZ8V2",
  "total_requests": 1,
  "ip_list": [
  {
  "ip": "100.64.33.183",
  "total_requests": 1,
  "devices": {
  "WINDOWS_7": 1
  },
  "methods": {
  "POST": 1
  },
  "urls": {
  "/apikeyheader/login": 1
  },
  "apis": {
  "apikeyheader": 1
  }
  }
  ]
  }
  ]
}
```


OAuth2 Token Metrics REST API

The OAuth2 token-based API is used to fetch the metrics for OAuth2 token across all application programming interface (API).

`Method: GET`

`URL: /v4/abs/oauthtokens?later_date=<yy-mm-dd>T<hh:mm>&earlier_date==<yy-mm-dd>T<hh:mm>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*

Copyright © 2025 Ping Identity Corporation

	Header	Value
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "oauth_token_metrics",
 "description": "This report contains a summary and detailed oauth token
  metrics across all APIs",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "summary": {
 "tokens": 30,
 "total_requests": 163250
 },
 "details": [
 {
 "token": "token_highresptime",
 "total_requests": 2,
 "ip_list": [
 {
 "ip": "127.0.0.1",
 "total_requests": 2,
 "devices": {
 "UNKNOWN": 2
 },
 "methods": {
 "GET": 2
 },
 "urls": {
 "/2_atm_app_oauth/longresponse": 1,
 "/atm_app_oauth/longresponse": 1
 },
 "apis": {
 "atm_app_oauth": 1,
 "2_atm_app_oauth": 1
 }
 }
 ]
 },
 {
 "token": "token10",
 "total_requests": 4596,
 "ip_list": [
 {
 "ip": "127.0.0.1",
 "total_requests": 4596,
 "devices": {
 "UNKNOWN": 4596
 },
 "methods": {
 "DELETE": 148,
 "POST": 1036,
 "GET": 1796,
 "PUT": 1616
 },
 "urls": {
 "/2_atm_app_oauth/put200": 656,
 "/atm_app_oauth/delete200": 68,
 "/2_atm_app_oauth/put400": 152,
 "/atm_app_oauth/delete400": 6
 },
 "apis": {
 "atm_app_oauth": 2298,
```


Copyright © 2025 Ping Identity Corporation

```
"2_atm_app_oauth": 2298
}
}
]
},
{
"token": "token14",
"total_requests": 7604,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 7604,
"devices": {
"UNKNOWN": 7604
},
"methods": {
"DELETE": 1596,
"POST": 160,
"GET": 4000,
"PUT": 1848
},
"urls": {
"/2_atm_app_oauth/put200": 846,
"/atm_app_oauth/delete200": 742,
"/2_atm_app_oauth/put400": 78,
"/2_atm_app_oauth/get400": 264
 },
"apis": {
"atm_app_oauth": 3802,
"2_atm_app_oauth": 3802
}
}
]
},
{
"token": "token_type2memory",
"total_requests": 2,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 2,
"devices": {
"UNKNOWN": 2
},
"methods": {
"POST": 2
},
"urls": {
"/2_atm_app_oauth/post200": 1,
"/atm_app_oauth/post200": 1
},
"apis": {
"atm_app_oauth": 1,
"2_atm_app_oauth": 1
}
}
]
},
{
"token": "token_method",
"total_requests": 2,
"ip_list": [
```


Copyright © 2025 Ping Identity Corporation

```
  {
  "ip": "127.0.0.1",
  "total_requests": 2,
  "devices": {
  "UNKNOWN": 2
  },
  "methods": {
  "HEAD": 2
  },
  "urls": {
  "/2_atm_app_oauth/get400": 1,
  "/atm_app_oauth/get400": 1
  },
  "apis": {
  "atm_app_oauth": 1,
  "2_atm_app_oauth": 1
  }
  }
  ]
  }
  ]
}
```


Username Metrics REST API

The Username base Metrics application programming interface (API) is used to fetch the metrics for username across all APIs.

`Method: GET`

`URL: /v4/abs/username?later_date=<yy-mm-dd>T<hh:mm>&earlier_date==<yy-mm-dd>T<hh:mm>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "username_metrics",
    "description": "This report contains a summary and detailed username metrics across all APIs",
    "earlier_date": "Wed May 22 12:00:00:000 2019",
    "later_date": "Fri Jun 28 12:00:00:000 2019",
    "summary": {
        "usernames": 4,
        "total_requests": 700
    },
    "details": [
        {
            "username": "t4",
            "tokens": [
                "t4VjqtSC",
                "t4XjDKtD",
                "t4JGkNZO",
                "t4gTqCqM",
                "t4UTgLaK",
                "t4mhTDNj",
                "t4srzDrl"
            ],
            "total_requests": 70,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 35,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                },
                {
                    "ip": "127.0.0.1",
                    "total_requests": 35,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                }
            ]
        },
        {
            "username": "t7",
```


Copyright © 2025 Ping Identity Corporation

```
            "tokens": [
                "t7cnVFBi",
                "t7wGQSnc",
                "t7XnAlRa",
                "t7MYwQan",
                "t7jzNFVF",
                "t7nsdecG",
                "t7Datxrw"
            ],
            "total_requests": 70,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 35,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                },
                {
                    "ip": "127.0.0.1",
                    "total_requests": 35,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                }
            ]
        },
        {
            "username": "t0",
            "tokens": [
                "t0iPoYEc",
                "t0wkCuYC",
                "t0YXowow",
                "t0NSwIjU",
                "t0PRwPik",
                "t0tEtlzI",
                "t0XBLmcE"
            ],
            "total_requests": 70,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 35,
                    "devices": {
```


Copyright © 2025 Ping Identity Corporation

```
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                },
                {
                    "ip": "127.0.0.1",
                    "total_requests": 35,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                }
            ]
        },
        {
            "username": "t3",
            "tokens": [
                "t3GUUfmD",
                "t3tRVhdk",
                "t3nkCZIR",
                "t3EFpRTc",
                "t3PuDsBr",
                "t3xGzXXB",
                "t3pZoWgX"
            ],
            "total_requests": 70,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 35,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                },
                {
                    "ip": "127.0.0.1",
                    "total_requests": 35,
```


Copyright © 2025 Ping Identity Corporation

```
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                }
            ]
        }
    ]
}
```


Anomalies REST API

The Anomalies application programming interface (API) is used to fetch the list of anomalies.

The response contains anomalies count for the API, request success or failure count, and so on.

`Method: GET`

`URL: /v4/abs/anomalies?later_date=<>earlier_date=<>&api=<api_name>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_anomalies",
 "description": "This report contains information on anomalous activity
  on the specified API.",
 "earlier_date": "Sun Jan 12 18:00:00:000 2018",
 "later_date": "Tue Jan 14 18:00:00:000 2018",
 "api_name": "shop",
 "anomalies_summary": {
 "api_url": "shopapi",
 "total_anomalies": 14,
 "most_suspicious_ips": [],
 "most_suspicious_anomalies_urls": []
 },
 "anomalies_details": {
 "url_anomalies": {
 "suspicious_sessions": [],
 "suspicious_requests": []
 },
 "ioc_anomalies": [
 {
 "anomaly_type": "API Memory Attack Type 2",
 "cookies": [
 {
 "cookie": "AMAT_2_H",
 "access_time": [
 "Mon Jan 13 01:01:33:589 2018"
 ]
 },
 {
 "cookie": "AMAT_2_H",
 "access_time": [
 "Mon Jan 13 01:01:33:589 2018"
 ]
 }
 ]
 },
 {
 "anomaly_type": "Data Exfiltration Attack",
 "cookies": [
 {
 "cookie": "data_exfilteration_VH",
 "access_time": [
 "Mon Jan 13 04:54:49:222 2018"
 ]
 },
 {
 "cookie": "data_exfilteration_H",
 "access_time": [
 "Mon Jan 13 05:26:53:981 2018"
 ]
 }
 ]
 },
 {
 "anomaly_type": "Cookie DoS Attack",
 "cookies": [
 {
 "cookie": "data_exfilteration_VH",
 "access_time": [
```


Copyright © 2025 Ping Identity Corporation

```
"Mon Jan 13 04:54:49:222 2018"
]
},
{
"cookie": "AMAT_1_freq_VH",
"access_time": [
"Sun Jan 12 23:17:55:931 2018"
]
},
{
"cookie": "data_exfilterationHH",
"access_time": [
"Mon Jan 13 05:39:18:515 2018"
]
},
{
"cookie": "AMAT_2_VH",
"access_time": [
"Sun Jan 12 23:59:39:483 2018"
]
}
]
},
{
"anomaly_type": "Extreme Client Activity Attack",
"cookies": [
{
"cookie": "data_exfilteration_VH",
"access_time": [
"Mon Jan 13 04:54:49:222 2018"
]
},
{
"cookie": "AMAT_1_VH",
"access_time": [
"Sun Jan 12 23:17:55:931 2018"
]
},
{
"cookie": "data_exfilteration_H_H",
"access_time": [
"Mon Jan 13 05:39:18:515 2018"
]
},
{
"cookie": "AMAT_2_VH",
"access_time": [
"Sun Jan 12 23:59:39:483 2018"
]
}
]
}
]
}
}
```


Anomalies across APIs

The across application programming interface (API) Anomalies REST API is used to fetch the list of anomalies.

Copyright © 2025 Ping Identity Corporation

The response contains the type of anomalies, the type ID and the date range when the anomaly was detected.

`Method: GET`

`URL: /v4/abs/anomalies?later_date=<>earlier_date=<>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
[
    {
        "company": "ping identity",
        "anomaly_type": "Stolen API Key Attack - Per API Key",
        "type": 31,
        "name": "api_anomaly_type",
        "description": "Client (API Key) reusing API Keys to deceive application services",
        "earlier_date": "Wed May 22 12:00:00:000 2019",
        "later_date": "Fri Jun 28 12:00:00:000 2019",
        "api_name": "all"
    },
    {
        "company": "ping identity",
        "anomaly_type": "Probing Replay Attack - API Key",
        "type": 32,
        "name": "api_anomaly_type",
        "description": "Probing or breach attempts on an API service – also called fuzzing",
        "earlier_date": "Wed May 22 12:00:00:000 2019",
        "later_date": "Fri Jun 28 12:00:00:000 2019",
        "api_name": "all"
    },
    {
        "company": "ping identity",
        "anomaly_type": "Extended Probing Replay Attack - API key",
        "type": 33,
        "name": "api_anomaly_type",
        "description": "Probing or breach attempts on an API service – also called fuzzing",
        "earlier_date": "Wed May 22 12:00:00:000 2019",
        "later_date": "Fri Jun 28 12:00:00:000 2019",
        "api_name": "all"
    },
    {
        "company": "ping identity",
        "anomaly_type": "Account Takeover Attack Type 1 - Username",
        "type": 34,
        "name": "api_anomaly_type",
        "description": "Abnormal activity by user indicating his/her credentials are compromised",
        "earlier_date": "Wed May 22 12:00:00:000 2019",
        "later_date": "Fri Jun 28 12:00:00:000 2019",
        "api_name": "all"
    },
    {
        "company": "ping identity",
        "anomaly_type": "Account Takeover Attack Type 2 - Username",
        "type": 35,
        "name": "api_anomaly_type",
        "description": "Abnormal activity by user indicating his/her credentials are compromised",
        "earlier_date": "Wed May 22 12:00:00:000 2019",
        "later_date": "Fri Jun 28 12:00:00:000 2019",
        "api_name": "all"
    },
    {
        "company": "ping identity",
        "anomaly_type": "Sequence Attack",
```


Copyright © 2025 Ping Identity Corporation

```
        "type": 36,
        "name": "api_anomaly_type",
        "description": "Abnormal sequence of transactions",
        "earlier_date": "Wed May 22 12:00:00:000 2019",
        "later_date": "Fri Jun 28 12:00:00:000 2019",
        "api_name": "all"
    }
]
```

OAuth2 Token Forensics REST API

The OAuth2 token forensics provides information like total number of requests for a token and the number of attacks identified using the token.

`Method: GET`

`URL: /v4/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&token=<oauth2_token>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_token",
 "description": "This report contains a summary and detailed information on
  metrics, attacks and anomalies for the specified token across all APIs.",
 "earlier_date": "Tue Feb 13 18:00:00:000 2018",
 "later_date": "Sun Feb 18 18:00:00:000 2018",
 "summary": {
 "total_requests": 6556,
 "total_attacks": 2,
 "total_anomalies": 0
 },
 "details": {
 "metrics": {
 "token": "token1",
 "total_requests": 6556,
 "ip_list": [
 {
 "ip": "127.0.0.1",
 "total_requests": 6556,
 "devices": {
 "UNKNOWN": 6556
 },
 "methods": {
 "DELETE": 472,
 "POST": 140,
 "GET": 1944,
 "PUT": 4000
 },
 "urls": {
 "/atm_app_oauth/delete200": 218,
 "/atm_app_oauth/get200": 850,
 "/atm_app_oauth/post400": 8,
 "/atm_app_oauth/post200": 62,
 "/atm_app_oauth/put400": 62,
 "/atm_app_oauth/get400": 122,
 "/atm_app_oauth/put200": 1938,
 "/atm_app_oauth/delete400": 18,
 "/2_atm_app_oauth/put200": 1938,
 "/2_atm_app_oauth/post200": 62,
 "/2_atm_app_oauth/delete200": 218,
 "/2_atm_app_oauth/delete400": 18,
 "/2_atm_app_oauth/put400": 62,
 "/2_atm_app_oauth/post400": 8,
 "/2_atm_app_oauth/get400": 122,
 "/2_atm_app_oauth/get200": 850
 },
 "apis": {
 "atm_app_oauth": 3278,
 "2_atm_app_oauth": 3278
 }
 }
 ]
 },
 "attack_types": {
 "API Memory Attack Type 1": [
 "atm_app_oauth",
 "2_atm_app_oauth"
 ],
 "Data Poisoning Attack": [
```


Copyright © 2025 Ping Identity Corporation

```
    "atm_app_oauth",
    "2_atm_app_oauth"
    ]
    },
    "anomaly_types": {}
    }
  }
```

IP Forensics REST API

The Internet Protocol (IP) forensics application programming interface (API) provides forensics information for an IP address during a specified period.

Information delivered includes attack types, metrics, and anomaly details.

`Method: GET`

`URL: /v4/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&IP=<IP_address>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_ip",
 "description": " This report contains a summary and detailed information
  on all attacks, metrics, and anomalies for the specified IP address on
  the defined API.",
 "summary": {
 "total_requests": 18222,
 "total_ioctypes": 0,
 "total_anomalies": 0
 },
 "details": {
 "ioc_types": [],
 "metrics": {
 "no_session": [
 {
 "start_time": "Sat Jan 04 15:30:00:000 2018",
 "end_time": "Sat Jan 04 15:39:59:952 2018",
 "total_requests": 2749,
 "source_ip": "100.64.10.203",
 "path": "/atmapp/login"
 "methods": [
 "GET"
 ]
 },
 {
 "start_time": "Sat Jan 04 15:30:00:000 2018",
 "end_time": "Sat Jan 04 15:39:59:952 2018",
 "total_requests": 2952,
 "source_ip": "100.64.10.203",
 "path": "/atmapp/upload"
 },
 {
 "start_time": "Sat Jan 04 15:30:00:000 2018",
 "end_time": "Sat Jan 04 15:39:59:952 2018",
 "total_requests": 9547,
 "source_ip": "100.64.10.203",
 "path": "/atmapp/zipcode"
 },
 {
 "start_time": "Sat Jan 04 15:30:00:000 2018",
 "end_time": "Sat Jan 04 15:39:59:952 2018",
 "total_requests": 2964,
 "source_ip": "100.64.10.203",
 "path": "/atmapp/update"
 }
 ],
 "session": [
 {
 "session_id": "ZP7FE32357SPVT5X",
 "start_time": "Sat Jan 04 15:35:14:241 2018",
 "end_time": "Sat Jan 04 15:35:14:241 2018",
 "total_requests": 1,
 "source_ip": [
 {
 "ip": "100.64.10.203",
 "count": 1,
 "method": [
 "POST"
 ]
```


Copyright © 2025 Ping Identity Corporation

```
  }
  ],
  "user_agent": [
  {
  "user_agent": "IE11",
  "count": 1
  }
  ],
  "path_info": [
  {
  "path": "/atmapp/upload",
  "count": 1
  }
  ],
  "device": [
  {
  "device": "WINDOWS_7",
  "count": 1
  }
  ]
  },

  "device": [
  {
  "device": "MAC_OS_X",
  "count": 1
  }
  ]
  },

  "start_time": "Sat Jan 04 15:40:00:000 2018",
  "end_time": "Sat Jan 04 15:30:00:000 2018",
  "api_name": "atmapp"
 }
```


Cookie Forensics REST API

Cookie forensics application programming interface (API) provides forensics information for a cookie during a specified period.

Information provided includes attack types, metrics, and anomaly details.

`Method: GET`

`URL: /v4/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm> &cookie=<cookie_value>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_abs_cookie",
 "description": "This report contains a summary and detailed information
  on all attacks, metrics, and anomalies for the specified cookie on
  the defined API",
 "earlier_date": "Mon Jan 17 06:40:00:000 2018",
 "later_date": "Mon Jan 17 07:00:00:000 2018",
 "api_name": "shop",
 "summary": {
 "total_requests": 501,
 "total_anomalies": 0,
 "total_ioc": 3
 },
 "details": {
 "ioc_types": [
 "data_exfiltration_attack",
 "cookie_dos_attack",
 "extreme_client_activity_attack"
 ],
 "metrics": [
 {
 "session_id": "extreme_client_activity_500_request",
 "start_time": "Mon Jan 17 06:47:19:687 2018",
 "end_time": "Mon Jan 17 06:47:20:505 2018",
 "total_requests": 501,
 "source_ip": [
 {
 "ip": "100.100.10.12",
 "count": 501,
 "method": [
 "POST",
 "GET"
 ]
 }
 ],
 "user_agent": [
 {
 "user_agent": "CHROME",
 "count": 501
 }
 ],
 "path_info": [
 {
 "path": "/shopapi/get",
 "count": 500
 },
 {
 "path": "/shopapi/login",
 "count": 1
 }
 ],
 "device": [
 {
 "device": "LINUX",
 "count": 501
 }
 ]
```


Copyright © 2025 Ping Identity Corporation

```
  }
  ],
  "anomalies": []
  }
 }
```

Token Forensics REST API

Token forensics application programming interface (API) provides forensics information for a token during a specified period.

Information provided includes attack types, metrics, and anomaly details.

`Method: GET`

`URL: /v4/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm> &token=<oauth2_token>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_abs_token",
    "description": "This report contains a summary and detailed information on metrics, attacks and anomalies for the
specified token across all APIs.",
    "earlier_date": "Wed May 22 12:00:00:000 2019",
    "later_date": "Fri Jun 28 12:00:00:000 2019",
    "summary": {
        "total_requests": 10,
        "total_attacks": 0,
        "total_anomalies": 0
    },
    "details": {
        "metrics": {
            "token": "t3nkCZIR",
            "total_requests": 10,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 5,
                    "devices": {
                        "LINUX": 5
                    },
                    "methods": {
                        "POST": 5
                    },
                    "urls": {
                        "/atm_app_oauth": 5
                    },
                    "apis": {
                        "atm_app_oauth": 5
                    }
                },
                {
                    "ip": "127.0.0.1",
                    "total_requests": 5,
                    "devices": {
                        "LINUX": 5
                    },
                    "methods": {
                        "POST": 5
                    },
                    "urls": {
                        "/atm_app_oauth": 5
                    },
                    "apis": {
                        "atm_app_oauth": 5
                    }
                }
            ]
        },
        "attack_types": {},
        "anomaly_types": {}
    }
}
```


Copyright © 2025 Ping Identity Corporation

API Key Forensics REST API

API Key forensics application programming interface (API) provides forensics information for a API Key during a specified period.

Information provided includes attack types, metrics, and anomaly details.

`Method: GET`

`URL: /v4/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm> &api_key=<api_key>`

	Header	Value
Access Key	`x-abs-ak`	`<string>`
Secret Key	`x-abs-sk`	`<string>`

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_abs_api_key",
    "description": "This report contains a summary and detailed information on metrics, attacks and anomalies for the
specified api key across all APIs.",
    "earlier_date": "Sat Jan 12 13:30:00:000 2019",
    "later_date": "Tue Dec 31 18:00:00:000 2019",
    "summary": {
        "total_requests": 2621,
        "total_attacks": 1,
        "total_anomalies": 1
    },
    "details": {
        "metrics": {
            "api_key": "finite_api_key",
            "total_requests": 2621,
            "ip_list": [
                {
                    "ip": "192.168.2.2",
                    "total_requests": 457,
                    "devices": {
                        "UNKNOWN": 457
                    },
                    "methods": {
                        "GET": 457
                    },
                    "urls": {
                        "/atm_app/getzipcode": 457
                    },
                    "apis": {
                        "atm_app": 457
                    }
                },
                {
                    "ip": "192.168.2.1",
                    "total_requests": 560,
                    "devices": {
                        "UNKNOWN": 560
                    },
                    "methods": {
                        "GET": 560
                    },
                    "urls": {
                        "/atm_app/getzipcode": 560
                    },
                    "apis": {
                        "atm_app": 560
                    }
                },
                {
                    "ip": "192.168.2.3",
                    "total_requests": 404,
                    "devices": {
                        "UNKNOWN": 404
                    },
                    "methods": {
                        "GET": 404
                    },
                    "urls": {
                        "/atm_app/getzipcode": 404
```


Copyright © 2025 Ping Identity Corporation

```
                    },
                    "apis": {
                        "atm_app": 404
                    }
                },
                {
                    "ip": "192.168.2.5",
                    "total_requests": 1200,
                    "devices": {
                        "UNKNOWN": 1200
                    },
                    "methods": {
                        "GET": 1200
                    },
                    "urls": {
                        "/atm_app/getzipcode": 1200
                    },
                    "apis": {
                        "atm_app": 1200
                    }
                }
            ]
        },
        "attack_types": {
            "Stolen API Key Attack- Per API Key": [
                "all"
            ]
        },
        "anomaly_types": {
            "Stolen API Key Attack- Per API Key": [
                "all"
            ]
        }
    }
}
```


Username Forensics REST API

Username forensics application programming interface (API) provides forensics information for a username during a specified period.

Information provided includes attack types, metrics, and anomaly details.

`Method: GET`

`URL: /v4/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm> &username=<username>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
    "company": "ping identity",
    "name": "api_abs_username",
    "description": "This report contains a summary and detailed information on metrics, attacks and anomalies for the
specified user name across all APIs.",
    "earlier_date": "Sat Jan 12 13:30:00:000 2019",
    "later_date": "Tue Dec 31 18:00:00:000 2019",
    "summary": {
        "total_requests": 109965,
        "total_attacks": 0,
        "total_anomalies": 0
    },
    "details": {
        "metrics": {
            "username": "t4",
            "tokens": [
                "t4MFBkEe",
                "t4GpEkUS",
                "t4ZxUOjb",
                "t4QEvJKT"
            ],
            "total_requests": 109965,
            "ip_list": [
                {
                    "ip": "127.0.0.28",
                    "total_requests": 54983,
                    "devices": {
                        "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110
Safari/537.36": 54983
                    },
                    "methods": {
                        "POST": 54983
                    },
                    "urls": {
                        "/atm_app_oauth": 54983
                    },
                    "apis": {
                        "atm_app_oauth": 54983
                    }
                },
                {
                    "ip": "127.0.0.1",
                    "total_requests": 54982,
                    "devices": {
                        "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110
Safari/537.36": 54982
                    },
                    "methods": {
                        "POST": 54982
                    },
                    "urls": {
                        "/atm_app_oauth": 54982
                    },
                    "apis": {
                        "atm_app_oauth": 54982
                    }
                }
            ]
```


Copyright © 2025 Ping Identity Corporation

```
        },
        "attack_types": {},
        "anomaly_types": {}
    }
}
```

Attack Types REST and WebSocket APIs

The Attack Type application programming interface (API) lists attack details based on the attack ID provided in the API query parameter.

The attack type ID ranges from 1-37 for REST APIs and 50-53 for WebSocket APIs. The REST API attacks can be per API or across APIs.

`Method: GET`

`URL for per API attacks (REST and WebSocket): /v4/abs/attack?later_date<>&earlier_date<>&api=<api_name>&type=<type_id>`

`URL for across API attacks: /v4/abs/attack?later_date<>&earlier_date<>&type=<type_id>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample response:

```
{
 "company": "ping identity",
 "description": " Client (IP or Cookie) extracting an abnormal amount of
  data for given API",
 "earlier_date": "Sat Jun 01 08:20:00:000 2019",
 "later_date": "Wed Jun 05 13:20:00:000 2019",
 "api_name": "atmapp",
 "ioc_type": "Data Exfiltration",
 "ips": [
 {
 "ip": "100.64.6.50",
 "access_time": [
 "Tue Jun 04 16:09:59:935 2019"
 ]
 },
 {
 "ip": "100.64.6.51",
 "access_time": [
 "Tue Jun 04 16:09:59:935 2019",
 "Tue Jun 04 16:39:59:996 2019"
 ]
 }
 ]
}
```

Copyright © 2025 Ping Identity Corporation

Flow Control REST API

The Flow Control application programming interface (API) is used to fetch details of all connections that exceeded the threshold value for client spike, server spike, connection queued, connection rejected, bytes-in spike, and bytes-out spike.

> **Note**
>
> The flow control report is only available when API Security Enforcer (ASE) is deployed in inline mode.

`Method: GET`

`URL: /v4/abs/flowcontrol?later_date=<>&earlier_date=<>&api=<api_name>`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_flowcontrol",
 "description": "This report contains flow control information for the
  specified API.",
 "earlier_date": "Wed Jan 01 08:20:00:000 2018",
 "later_date": "Sun Jan 05 13:20:00:000 2018",
 "api_name": "websocket",
 "summary": {
 "client_spike": 610,
 "connection_queued": 0,
 "connection_quota_exceeded": 0,
 "bytes_in_spike": 2743,
 "bytes_out_spike": 287
 },
 "details": {
 "client_spike": [],
 "server_spike": [
 {
 "request_time": "Fri Jan 09 17:19:55:977 2016",
 "connection_id": "147378243",
 "source_ip": "100.64.26.163",
 "destination_api": "/atmapp/login"
 },
 {
 "request_time": "Fri Jan 09 17:19:55:991 2016",
 "connection_id": "1919058221",
 "source_ip": "100.64.20.230",
 "destination_api": "/atmapp/zipcode"
 }
 ],
 "connections_queued": [],
 "connections_rejected": [],
 "bytes_in_spike": [],
 "bytes_out_spike": []
 }
}
```


Blocked Connection REST API

The Blocked Connection application programming interface (API) is used to fetch the list of blocked or dropped connections.

The response includes anomalies count for the given API, such as request success or failure count.

`Method: GET`

`URL : /v4/abs/bc?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&details=true`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "earlier_date": "Wed Jan 01 08:20:00:000 2018",
 "later_date": "Sun Jan 05 13:20:00:000 2018",
 "api_blocked_connections": [
 {
 "date": "05September2016",
 "blocked_connections": [
 {
 "apiproxy_node":"204101a4-8b70-489d-98e9-
 aa3f6e67a93f",
 "blocked_connections": [
 {
 "category": "ioc",
 "details": []
 },
 {
 "category": "api",
 "details": [
 {
 "source": "100.64.31.235",
 "type": "no_backend_available",
 "destination_api": "/atmapp/zipcode"
 },
 {
 "source": "100.64.25.184",
 "type": "no_backend_available",
 "destination_api": "/atmapp/zipcode"
 },
 {
 "source": "100.64.6.137",
 "type": "no_backend_available",
 "destination_api": "/atmapp/zipcode"
 },
 {
 "source": "100.64.1.251",
 "type": "no_backend_available",
 "destination_api": "/atmapp/zipcode"
 }
 ]
 }
 ]
 }
 ]
 }
 ]
}
```


Backend Error REST API

The Backend Error application programming interface (API) displays errors reported by the backend servers.

`Method: GET`

`URL: /v4/abs/be?ealier_date=<>T<hh:mm>&later_date=<>T<hh:mm>&api=<api_name>`

Copyright © 2025 Ping Identity Corporation

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_backend_errors",
 "description": "This report contains details of backend error
  codes for the specified API",
 "earlier_date": "Wed Jan 01 08:20:00:000 2018",
 "later_date": "Sun Jan 05 13:20:00:000 2018",
 "api_name": "atmapp",
 "backend_error_summary": [
 {
 "error_code": "403",
 "error": "Forbidden",
 "count": 0
 },
 {
 "error_code": "404",
 "error": "Not Found",
 "count": 0
 },
 {
 "error_code": "500",
 "error": "Internal Server Error",
 "count": 16
 },
 {
 "error_code": "503",
 "error": "Service Unavailable",
 "count": 0
 },
 {
 "error_code": "504",
 "error": "Gateway Timeout",
 "count": 0
 }
 ],
 "backend_error_details": [
 {
 "error_code": "403",
 "details": []
 },
 {
 "error_code": "404",
 "details": []
 },
 {
 "error_code": "500",
 "details": [
 {
 "server": "192.168.11.164:3001",
 "request_url": "/atmapp/zipcode",
 "request_ip": "100.64.5.183:24078",
 "request_cookie": ""
 },
 {
 "server": "192.168.11.164:3002",
 "request_url": "/atmapp/zipcode",
 "request_ip": "100.64.18.126:61932",
 "request_cookie": ""
 },
 {
```


Copyright © 2025 Ping Identity Corporation

```
  "server": "192.168.11.164:3004",
  "request_url": "/atmapp/zipcode",
  "request_ip": "100.64.27.176:2908",
  "request_cookie": "JSESSIONID=6UQANJWB42U4A4PF"
  },
  {
  "server": "192.168.11.164:3004",
  "request_url": "/atmapp/zipcode",
  "request_ip": "100.64.14.237:21973",
  "request_cookie": "JSESSIONID=LJ66P3NQW5SDVW8Q"
  },
  {
  "server": "192.168.11.164:3003",
  "request_url": "/atmapp/zipcode",
  "request_ip": "100.64.5.101:5523",
  "request_cookie": ""
  },
  {
  "server": "192.168.11.164:3003",
  "request_url": "/atmapp/zipcode",
  "request_ip": "100.64.23.132:14473",
  "request_cookie": "JSESSIONID=NCTZ4RSOZP2IT2OU"
  },
  {
  "server": "192.168.11.164:3003",
  "request_url": "/atmapp/zipcode",
  "request_ip": "100.64.5.197:50811",
  "request_cookie": ""
  },
  {
  "server": "192.168.11.164:3003",
  "request_url": "/atmapp/zipcode",
  "request_ip": "100.64.26.70:49425",
  "request_cookie": ""
  }
  ]
  },
  {
  "error_code": "503",
  "details": []
  },
  {
  "error_code": "504",
  "details": []
  }
  ]
 }
```


List Valid URLs REST API

The List Valid Uniform Resource Locator (URL) application programming interface (API) provides information on all the URLs for the API.

The API reports the allowed methods and the count of number of times each URL has been accessed.

`Method: GET`

`URL: /v4/abs/validurl?api=<api_name>`

Copyright © 2025 Ping Identity Corporation

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "name": "api_url_list",
 "description": "This report provides information on access to each
  unique URL for the specified API",
 "api_name": "shop",
 "host_name": "app",
 "api_url": "shopapi",
 "allowed_methods": [
 "GET",
 "PUT",
 "POST",
 "DELETE",
 "HEAD"
 ],
 "url_list": [
 {
 "protocol": "HTTP/1.1",
 "urls": [
 {
 "url": "/shopapi/get_delay",
 "total_count": 11,
 "methods": [
 {
 "method": "GET",
 "count": 11
 }
 ]
 },
 {
 "url": "/shopapi/post",
 "total_count": 62109,
 "methods": [
 {
 "method": "POST",
 "count": 62109
 }
 ]
 },
 {
 "url": "/shopapi/get_mb",
 "total_count": 2,
 "methods": [
 {
 "method": "GET",
 "count": 2
 }
 ]
 },
 {
 "url": "/shopapi/login",
 "total_count": 2686,
 "methods": [
 {
 "method": "POST",
 "count": 2686
 }
 ]
 },
 {
```


Copyright © 2025 Ping Identity Corporation

```
"url": "/shopapi/get?dyanmic_cookie",
"total_count": 378,
"methods": [
{
"method": "GET",
"count": 378
}
]
},
{
"url": "/shopapi/logout",
"total_count": 16964,
"methods": [
{
"method": "POST",
"count": 16964
}
]
},
{
"url": "/shopapi/get?passwd",
"total_count": 1,
"methods": [
{
"method": "GET",
"count": 1
}
]
},
{
"url": "/shopapi/put",
"total_count": 62060,
"methods": [
{
"method": "PUT",
"count": 62060
}
]
}
]
} ] }
```


List Hacker's URL REST API

The List Invalid Uniform Resource Locator (URL) application programming interface (API) provides information on all invalid URLs accessed for an API.

The four types of invalid URLs are:

- **Irregular URL**
- **System Commands**
- **SQL Injection, and**
- **Buffer Overflow**

`Method: GET`

Copyright © 2025 Ping Identity Corporation

`URL: /v4/abs/hackersurl?api=<api_name>&earlier_date=””&later_date=””`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Response:

Copyright © 2025 Ping Identity Corporation

```
{
 "company": "ping identity",
 "description": "This report contains list of hackers URL for given API",
 "name": "api_hackers_url",
 "api_name": "universal_api",
 "invalid_urls": [
 {
 "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,-- ",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL,NULL,NULL#",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=(SELECT 46 FROM(SELECT COUNT(\*),CONCAT(0x717a71,))",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL,NULL#",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc' UNION ALL SELECT NULL,NULL,NULL,NULL,,NULL",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc UNION ALL SELECT NULL,NULL,NULL,NULL,NULL#",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL-- ",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL-- ",
 "ips": [
 "127.0.0.1"
 ]
 },
```


Copyright © 2025 Ping Identity Corporation

```
 {
 "url": "/index.php?id=abc UNION ALL SELECT NULL,NULL,NULL,NULL,NULL#",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc%' UNION ALL SELECT NULL-- ",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc) UNION ALL SELECT NULL,NULL,NULL,NULL-- ",
 "ips": [
 "127.0.0.1"
 ]
 },
 {
 "url": "/index.php?id=abc' UNION ALL SELECT NULL,NULL,NULL-- ",
 "ips": [
 "127.0.0.1"
 ]
 }
 ]
}
```

Delete deny list REST API

The delete deny list REST application programming interface (API) deletes active deny lists in API Behavioral Security (ABS).

The API checks if the client identifier is present in the active list or not before deleting.

`Method: PUT`

`URL: /v4/abs/attacklist/`

	Header	Value
Access Key	`x-abs-ak`	*<string>*
Secret Key	`x-abs-sk`	*<string>*

Sample Request to the API:

```
{
      "ips": [],
      "cookies": {},
      "oauth_tokens": [],
      "api_keys": [],
      "usernames": [“user_70”]
}
```

Copyright © 2025 Ping Identity Corporation

Sample response from the API when the client identifiers from active blacklists are deleted:

```
{
    "message": "The following attacks have been removed:",
    "attacklist": {
        "ips": [],
        "cookies": {},
        "oauth_tokens": [
            "SYU4R2ZZN1IDYI0L"
        ],
        "api_keys": [],
        "usernames": []
    },
    "status_code": "SUCCESS"
}
```

Sample response from the API when the deletion fails:

```
{
    "status_code": "INVALID_JSON",
    "message": "Invalid json. Please ensure all input fields are present and have valid values"
}
```

ABS log messages

The following table lists the critical log messages from `abs.log` and `aad.log` file.

Log message	Description
Warning:-Maximum Transaction limit is reached for this month	This message is logged in `abs.log` when the transaction limit is reached for the allotted license usage. For more information, see Configuring and updating an ABS license
Warning:- Attempt to shutdown ABS from 127.0.0.1	This message is logged in `abs.log` when shutdown of API Behavorial Security (ABS) AI engine is initiated.
Warning:- Failed to delete IPs from IOCs - try again	This message is logged in `abs.log` when the Attack list REST API encounters an issues while deleting the IP address from the deny list.
Warning:- Failed to delete tokens from IOCs - try again	This message is logged in `abs.log` when the Attack list REST API encounters an issues while deleting the OAuth token from the deny list
Warning:- Failed to delete usernames from IOCs - try again	This message is logged in `abs.log` when the Attack list REST API encounters an issues while deleting the usernames from the deny list.

Copyright © 2025 Ping Identity Corporation

Log message	Description
Warning:- Failed to delete API keys from IOCs - try again	This message is logged in `abs.log` when the Attack list REST API encounters an issues while deleting the API Keys from the deny list.
Warning:- License is Expired. Please renew your license	This message is logged in `abs.log` when ABS license has expired. For more information, see Configuring and updating an ABS license
Warning:- MongoDB primary node is down	This message is logged in `abs.log` when a MongoDB connection failure occurs.
Warning:- Stream init-wait interrupted	This message is logged in `abs.log` when streaming of access log files is interrupted.
Warning:- File system usage reached configured value of: 80% ABS will not accept new logs from ASE.	This message is logged in `abs.log` when ABS stops accepting access log files from ASE because of maximum use of filesystem.
Warning:- Error while closing mongo connections	This message is logged in `abs.log` when shutdown of MongoDB connection was not successful.
Warning:- Error while loading anomaly dictionary from mongo	This message is logged in `abs.log` when writing of anomalies to data directory fails.
Warning:- Error while closing file handle for stream config	This message is logged in `abs.log` when an error occurs while closing the streaming configuration file.
Error: exception while parsing license file `/opt/pingidentity/abs/config/PingIntelligence.lic`	This message is logged in `abs.log` when an error occurs while reading the license file. Add the file named `PingIntelligence.lic` to the specified path with read permission and restart the ABS AI engine
Error: License `/opt/pingidentity/abs/config/PingIntelligence.lic` is invalid. ABS will shut down now.	This message is logged in `abs.log` when an error is encountered while validating the license file. Provide a valid license file and restart the ABS AI engine
ABS will shut down now	This message is logged in `abs.log` when your free ABS license expires.
Attempting to initialize abs, but abs is already in *<message>*	This message is logged in `abs.log` when another ABS process is already running.

Copyright © 2025 Ping Identity Corporation

Log message	Description
error while loading `abs.properties` <*Custom run-time message*> The various custom error messages could be: • property <*abs_propertie*>is missing • invalid value for property log_level. Value should be string and member of [ALL,DEBUG,INFO,WARN,ERROR,FATAL,OFF] • property management_port is missing • invalid value for property management_port, value should be integer and (>=1 && ⇐65535) • invalid value for property jks_password, obfuscation of password failed. Please make sure you are using the correct `config/abs_master.key` file • invalid value for property jks_password, value should be obfuscated using the 'bin/cli.sh -u admin -p <password> obfuscate_keys' command • invalid value for property host_ip, value should be string and ipv4 address • property enable_emails is missing • invalid value for property smtp_host value should be string and should be as per rfc1024 and rfc1123	This message is logged in `abs.log` when: • Error occurs when `abs.properties` file is not configured with log_level specifications • Error occurs when `abs.properties` file is not configured with management_port specifications
error while loading `abs_resources.properties`	This message is logged in `abs.log` when `abs_resources.properties` doesn't contain values for memory and CPU parameters.
error while initializing mongodb replica set connections	This message is logged in `abs.log` when MongoDB initialization fails and cannot access a read or write client for connections.
error while reading enable_ssl key from mongo master	This message is logged in `abs.log` when MongoDB client tries to fetch the key from MongoDB collections.
error while reading root_api_attack key from mongo master	This message is logged in `abs.log` when MongoDB client tries to fetch the key from MongoDB collections.
error while reading `/config/abs.properties`	This message is logged in `abs.log` while loading and validating the `abs.properties` file. Check whether file exists and its permission.
invalid value for property jks_password, value should be obfuscated using the 'bin/cli.sh -u admin -p <*password*> obfuscate_keys' command	This message is logged in `abs.log` when an error occurs while deobfuscating the jks_password using the master_key.
error while loading auth keys from metadata db in mongo	This message is logged in `abs.log` when MongoDB is not accessible.

Copyright © 2025 Ping Identity Corporation

Log message	Description
error while loading restricted user auth keys from metadata db in mongo	This message is logged in `abs.log` when MongoDB is not accessible.
Unable to read `<abs_root_dir>/config/abs.jks` file	This message is logged in `abs.log` when `abs.jks` is not created properly or could not read the file or there is a permission issue.
error while starting management server *<runtime exception>*	This message is logged in `abs.log` when there is an issue when ABS starts.
API Behavioral Security stopped	This message is logged in `abs.log` when ABS is shut down.
MongoDB heartbeat failure	This message is logged in `abs.log` when ABS is unable to connect to MongoDB primary node.
ABS started successfully	This message is logged in `abs.log` when ABS starts.

Threshold range for Tn and Tx

The following table details the range of `Tn` and `Tx` for each attack type.

When manually adjusting the threshold values, the values must fall within the specified ranges.

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
REST API							
Data Exfiltration	1	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA	NA	NA
Single Client Login	2	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Multi Client Login	3	Tn = [1-100] Tx = “na”	NA	NA	NA	NA	NA
Stolen Cookie / Access Token	4	Tn = [2-10]	Tn = [1-19], Tx = [2-20]	NA	NA	NA	NA

Copyright © 2025 Ping Identity Corporation

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
API Memory Attack Type 1	5	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA	NA	NA
API Memory Attack Type 2	6	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA	NA	NA
Cookie DoS	7	Tn = [1-9] Tx = [2-10]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
API Probing Replay	8	Tn = [1-99] Tx = [2-100]	NA	NA	NA	NA	NA
API DoS Attack Type 1	9	Tn = [1-100] Tx = “[2-100]”	NA	NA	NA	NA	NA
Extreme Client Activity	10	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA	NA
Extreme App Activity	11	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA	NA
API DoS Attack	12	Tn = [1-100] Tx = “na”	NA	NA	NA	NA	NA
API DDoS Attack Type 2	13	NA	NA	NA	NA	NA	NA
Data Deletion	14	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA	NA	NA	NA
Data Poisoning	15	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	Tn = [1-32] Tx = [2-33]	NA	NA	NA

Copyright © 2025 Ping Identity Corporation

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
Stolen Token Attack Type 2	16	Tn = [2-10] Tx = “na”	Tn = [1-100]	Tn = [1-100]	NA	NA	NA
Stolen Cookie Attack Type 2	17	Tn = [2-10] Tx = “na”	Tn = [1-100]	Tn = [1-100]	NA	NA	NA
API Probing Replay Attack 2 (client identifier: cookie)	18	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
API Probing Replay Attack 2 (client identifier: token)	19	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
API Probing Replay Attack 2 (client identifier: IP address)	20	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Data Exfiltration Attack Type 2	21	Tn = [1-42] Tx = [2-43]	Tn = [0-30]	Tn = [1-100]	NA	NA	NA
Excessive Client Connections (client identifier : cookie)	22	Tn = [1-19], Tx =[2-20]	NA	NA	NA	NA	NA
Excessive Client Connections (client identifier : token)	23	Tn = [1-19], Tx =[2-20]	NA	NA	NA	NA	NA

Copyright © 2025 Ping Identity Corporation

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
Excessive Client Connections (client identifier : IP address)	24	Tn = [1-19], Tx =[2-20]	NA	NA	NA	NA	NA
Content Scraping Type 2	28	Tn = [1-29] Tx = [2-30]	Tn = [1-100]	NA	NA	NA	NA
Unauthorized client attack (client identifier: IP address)	29	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Single Client Login Attack Type 2 (client identifier: IP address)	30	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Stolen API Key Attack- API Key	31	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA
Probing Replay Attack - API Key	32	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA	NA	NA
Extended Probing Replay Attack - API Key	33	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA	NA	NA
User Probing Type 1	34	Tn = [1-99] Tx = [2-100]	Tn = [1-99] Tx = [2-100]	Tn = [1-9] Tx = [2-10]	Tn = [1-9] Tx = [2-20]	NA	NA
User Probing Type 2	35	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-29] Tx = [2-30]	NA	NA

Copyright © 2025 Ping Identity Corporation

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
Sequence attack	36	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA	NA
Header Manipulation	37	Tn = [1-99] Tx = [2-100]	Tn = [1-20] Tx = NA	Tn = [1-29] Tx = [2-30]	Tn = [1-100] Tx = NA	Tn = [1-2] Tx = NA	Tn = [1-100] Tx = NA
Account Takeover - UBA	38	Tn = [1-100] Tx = NA	Tn = [1-99] Tx = [2-100]	NA	NA	NA	NA
User Data Exfiltration Type 2	39	Tn = [1-32] Tx = [2-33]	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	NA	NA	NA
User Data Injection	40	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Query Manipulation Attack	41	Tn = [1-20] Tx = NA	Tn = [1-2] Tx = NA	Tn = [1-2] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-2] Tx = NA	Tn = [1-100] Tx = NA
Content Scraping Type 1	42	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA
WebSocket API							
WS Cookie Attack	50	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx= [2-20]	NA	NA	NA	NA
WS Identity Attack	51	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
WS DoS Attack	53	Tn = [1-100] Tx = “na”	NA	NA	NA	NA	NA

Copyright © 2025 Ping Identity Corporation

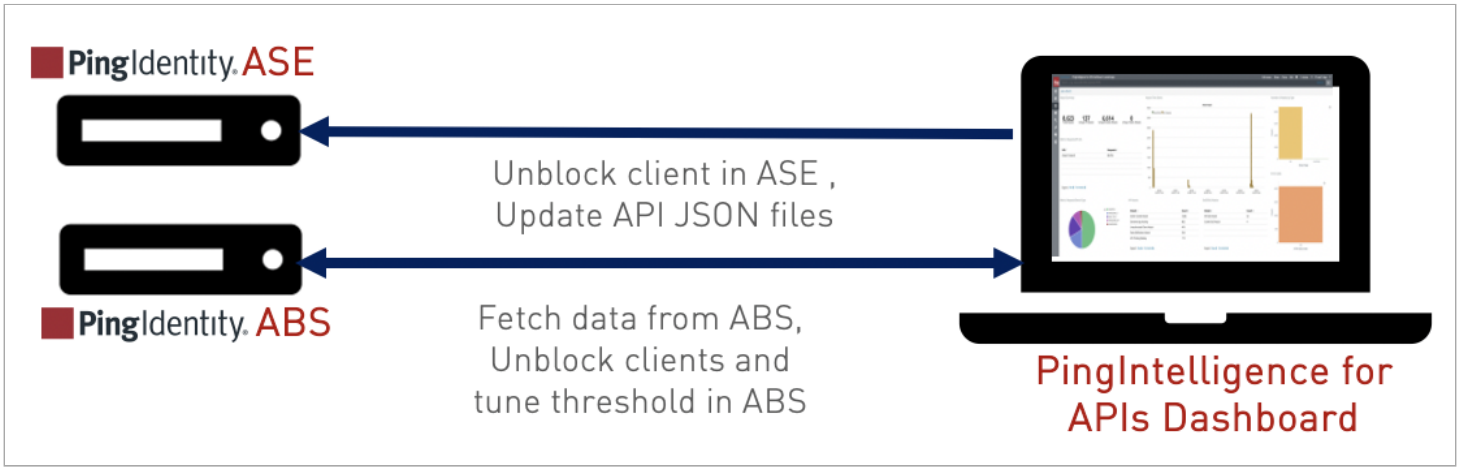
Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
WS Data Exfiltration Attack	54	Tn = [1-100] Tx = “na”	NA	NA	NA	NA	NA

PingIntelligence Dashboard

PingIntelligence for APIs Dashboard provides a graphical view of an API environment.

It provides insights on user activity, attack information, blocked connections, forensic data, and much more.

The following diagram shows the data flow between ASE, ABS AI Engine, and PingIntelligence Dashboard.



The Dashboard fetches data from the ABS AI engine and lets you:

- View API activity
- View the training status and other API information
- Organize APIs into logical groups
- Hide or display APIs from the main API dashboard, sort and search for APIs
- View the API dashboard
- View attack insight to understand why a client was flagged for an attack
- Unblock a client or tune AI engine thresholds
- Manage API Discovery using automatic or manual mode

Copyright © 2025 Ping Identity Corporation

Installing the PingIntelligence Dashboard

Learn how to install the PingIntelligence Dashboard.

Before you begin

Ensure that the following prerequisites are met:

- Server: 8 core CPU, 16 GB, 1 TB HDD
- Operating system: RHEL 7.9 or Ubuntu 18.04 LTS
- OpenJDK: 11.0.2
- SSL certificate: One private key and certificate. By default, PingIntelligence Dashboard uses the private key and certificate shipped with the binary.
- Password: If you want to change the default password, set a minimum 8 character password.
- API Behavioral Security (ABS): ABS URL, access, and secret key. Make sure that ABS is reachable from the PingIntelligence Dashboard machine.
- API Security Enforcer (ASE): ASE management URL, access, and secret key. Make sure that ASE is reachable from the PingIntelligence Dashboard machine.

> **Note**
>
> Connecting the Dashboard to ASE is optional. Functionality, such as adding discovered APIs to ASE and attack management, will be limited.

Make sure the following default port numbers are available:

- PingIntelligence Dashboard (WebGUI) server: 8030. Port number 8030 should be exposed to public internet. Make sure that your organization's firewall allows access to this port.
- Elasticsearch: 9200
- Dataengine: 8040
- H2 database: 9092. The H2 database is installed and runs as a part of the PingIntelligence Dashboard.

Make sure you have one of the following supported browsers installed. The following table shows the compatibility of PingIntelligence for APIs Dashboard with different browsers and their versions.

Operating System	Google Chrome	Mozilla Firefox	Apple Safari	Microsoft Edge
Mac OS Mojave -10.14	Version 56.0 and later	Version 69.0 and later	Version 12.0 and later	
Mac OS Sierra -10.12	Version 56.0 and later	Version 69.0 and later	Version 10.1 and later	
Mac OS High Sierra -10.13	Version 56.0 and later	Version 69.0 and later	Version 11.1 and later	

Copyright © 2025 Ping Identity Corporation

Operating System	Google Chrome	Mozilla Firefox	Apple Safari	Microsoft Edge
Mac OS Catalina -10.15	Version 56.0 and later	Version 69.0 and later	Version 13.0 and later	
Windows 8.1	Version56.0 and later	Version 69.0 and later		
Windows 10	Version 56.0 and later	Version 69.0 and later		Version 79.0 and later

Ensure you have completed the following configuration for the operating system:

- Increase the `ulimit` to 65536:

```
# sudo sysctl -w fs.file-max=65536
# sudo sysctl -p
```

- Increase the `vm.max_map_count` limit to 262144:

```
# sudo echo "vm.max_map_count=262144" >> /etc/sysctl.conf
# sudo sysctl -p
```

- Set `JAVA_HOME` to the *<jdk_install>* directory and add `<jdk_install>/bin` to the system `PATH` variable. *<jdk_install_dir>* is the directory where JDK is installed.
- Choose the *<pi_install_dir>* directory. The *<pi_install_dir>* directory is the directory where the PingIntelligence Dashboard is installed. This directory should be readable and writable by the logged in user.

About this task

Installing the PingIntelligence for APIs Dashboard automatically installs Elasticsearch.

There are two preconfigured login users in PingIntelligence Dashboard:

- `admin`
- `ping_user`

Multiple `admin` and `ping_user` can simultaneously sign on to PingIntelligence Dashboard. The admin user has full access toPingIntelligence Dashboard. An `admin` can view the dashboard of various APIs as well as tune threshold and unblock a client identifier. `ping_user` can only view the API dashboard. A total of 25 `admin` and `ping_user` can sign on simultaneously.

To install the PingIntelligence Dashboard:

Steps

1. Create a *<ping_install_dir>* directory on your host machine.

   Make sure that the user has read and write permissions for the *<ping_install_dir>* directory.

2. Download the PingIntelligence Dashboard binary.

   1. Under Download AI Engine and Tools, click Dashboard 5.1.0.1.

Copyright © 2025 Ping Identity Corporation

3. Download Elasticsearch 6.8.1 (macOS/RHEL).
4. Change the directory to *ping_install_dir*:

```
# cd pi_install_dir
```

5. Untar the PingIntelligence Dashboard:

```
# tar -zxf pi-api-dashboard-5.1.tar.gz
```

6. Add the MongoDB IP address in `webgui.properties`.
7. Copy the MongoDB certificate to the Dashboard virtual machine (VM).
8. Import the MongoDB certificate to `webgui.jks` using the following command:

```
keytool -import -keystore dataengine.jks -storetype JKS -storepass changeme -alias mongo -file
mongo.crt -noprompt
```

9. Change the directory to `pingidentity/webgui/`:

```
# cd pingidentity/webgui/
```

10. Install the PingIntelligence Dashboard by entering the following command and follow the instructions displayed on the prompt:

```
# ./bin/pi-install-ui.sh
```

Copyright © 2025 Ping Identity Corporation

```
# ./bin/pi-install-ui.sh

elasticsearch-7.13.4.tar.gz file path >
Use bundled ssl key and self signed certificate for ui server [y/n]?  >[y]
Use default password [changeme] for all components and users [y/n]?  >[y]
ABS url  >[https://127.0.0.1:8080]
ABS access key  >[abs_ak]
ABS secret key  >[abs_sk]
API Service URL  >[https://127.0.0.1:8050]
Kafka Host:Port >[127.0.0.1:9093]
Kafka Authentication username  >[pi4api_de_user]
Kafka Group ID  >[pi4api.data-engine]
ASE management url  >[]
extracting elasticsearch package
creating elasticsearch config keystore
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
Created elasticsearch keystore in <pi_install_dir>/pingidentity/elasticsearch/config/
elasticsearch.keystore
elasticsearch config keystore created
Generating a 2048 bit RSA private key
.........................................+++
.....................+++
writing new private key to 'config/ssl/autogen_es.key'
-----
creating password protected pkcs#12 keystore for elasticsearch private key and certificate
pkcs#12 keystore created at config/ssl/elastic-certificates.p12
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
configuring elasticsearch. Please wait 15 seconds
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will
likely be removed in a future release.
elasticsearch config is completed
configuring dataengine
configuring webgui
starting webgui for configuration update
WebGUI configured for UTC timezone.
WebGUI 5.1 starting...
please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for more details
success: password updated.
Note: All active sessions for this user are invalidated. Login with new credentials
success: password updated.
Note: All active sessions for this user are invalidated. Login with new credentials
WebGUI 5.1
WebGUI is stopped.
webgui configuration done

UI configuration done
writing internal credentials to <pi_install_dir>/pingidentity/webgui/install/webgui_internal.creds
Start UI [y/n]?  >[y]
starting elasticsearch...
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
```


Copyright © 2025 Ping Identity Corporation

```
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will
likely be removed in a future release.
elasticsearch started
starting dataengine
Data Engine configured for UTC timezone.
PingIntelligence Data Engine 5.1 starting...
Data-Engine started
starting webgui
WebGUI configured for UTC timezone.
WebGUI 5.1 starting...
please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for more details
Please access WebGUI at https://<pi_install_host>:8030

<pi_install_host> can be ip address, hostname or fully qualified domain name of this server.
<pi_install_host> should be reachable from your computer.

Credentials:
  1) Username: admin
     Password: changeme
  2) Username: ping_user
     Password: changeme

Important Actions:
1) Credentials for all internal components are available in <pi_install_dir>/pingidentity/webgui/
install/webgui_internal.creds file.
    Move this file from this server and securely keep it elsewhere.
    For any debugging purposes you will be asked to get credentials for a component from this file.
2) Following obfuscation master keys are auto-generated
      <pi_install_dir>/pingidentity/webgui/config/webgui_master.key
      <pi_install_dir>/pingidentity/dataengine/config/dataengine_master.key
```

Important

The `ASE management url` is an optional parameter.

11. Verify installation by checking the process IDs (`pid`) of each component in the following locations:
    1. Elasticsearch: `<pi_install_dir>/elasticsearch/logs/elasticsearch.pid`
    2. Dataengine: `<pi_install_dir>/dataengine/logs/dashboard.pid`
    3. Web GUI: `<pi_install_dir>/webgui/logs/webgui.pid`
12. Tune the Dashboard performance parameters by configuring the following three parameters for better performance.

Note

Note the following tuning parameters if you have your setup of Elasticsearch. If you have used PingIntelligence automated deployment or the `pi-install-ui.sh` script to deploy the Dashboard, the tuning of the parameters below is done as part of installation.

Copyright © 2025 Ping Identity Corporation

Parameter	Description	Location
Elasticsearch		
`-Xms` and `-Xmx`	◦ `-Xms` : Defines the minimum heap size of Elasticsearch. Set to 4 GB as `Xms4g` . ◦ `-Xmx` : Defines the maximum heap size of Elasticsearch. Set to 4 GB as `Xmx4g` .	`$ES_HOME/config/jvm.options`
`thread_pool.search.size`	Defines thread pool size for count/search/suggest operations in Elasticsearch. Configure to 50% of total CPUs allocated.	`$ES_HOME/config/elasticsearch.yml`

Troubleshooting:

To detect and mitigate attacks such as cross-site scripting (XSS), the PingIntelligence Dashboard implements Content Security Policy (CSP). The following are the configuration details:

```
Response header - Content-Security-Policy
```

```
Response header value - default-src 'self'; font-src 'self' use.typekit.net; script-src 'self'
use.typekit.net; style-src 'self' 'unsafe-inline' use.typekit.net p.typekit.net; img-src 'self'
data: p.typekit.net;
```

Accessing the PingIntelligence Dashboard

Access the PingIntelligence for APIs Dashboard from a browser at the default Uniform Resource Locator (URL): https://<*pi_install_host*>:8030.

About this task

There are two preconfigured login users in PingIntelligence for APIs Dashboard:

- `admin`
- `ping_user`

Multiple users can share the `admin` and `ping_user` logins simultaneously on PingIntelligence Dashboard. The admin user has access to all PingIntelligence Dashboard functions. A `ping_user` can only view the application programming interface (API) dashboards.

The PingIntelligence Dashboard is categorized into the following components:

Copyright © 2025 Ping Identity Corporation

Dashboard Component	Description
Main Dashboard	Available for `admin` and `ping_user`
APIs	Available only for `admin` user
Discovered APIs	Available only for `admin` user
Attack Management	Available only for `admin` user
License	Available only for `admin` user
Active Sessions	Available only for `admin` user
Settings	Available only for `admin` user

> **Note**
>
> For further information on dashboard features, usage, and administration, see PingIntelligence Dashboard.

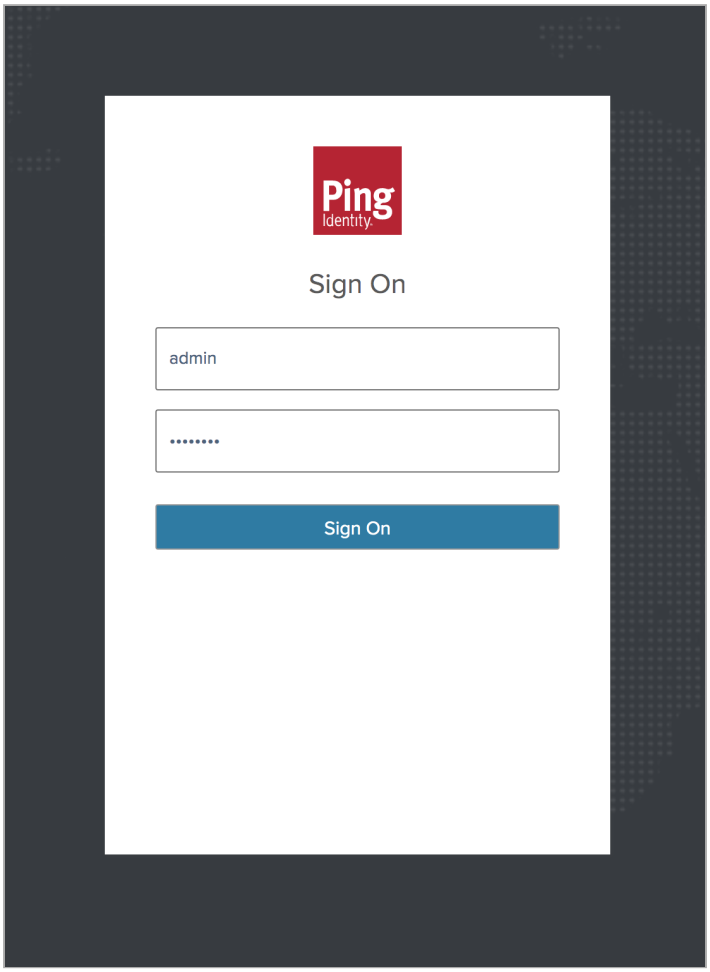
Steps

1. At the sign-on prompt, sign on as `admin` or `ping_user`.

   The default password for both the users is `changeme`.

   > **Caution**
   >
   > You must change the default password for production deployments. However, in a Docker Proof of Concept deployment, use the default password.

Copyright © 2025 Ping Identity Corporation



> **Note**
>
> If the Dashboard is not accessible, check if the default port (8030) was changed by your system administrator.

2. Optional: Change the password using the following command-line interface (CLI) command:

```
#  <pi_install_dir>/webgui/bin/cli.sh -u admin update_ui_password --username -value  <admin or
ping_user>  --new-password -p
Enter admin password >  <current admin password>
Enter new password >  <new password>
Reenter new password >  <new password>
success: password updated.
```

3. Optional: To configure the maximum number of active sessions, set the `pi.webgui.session.max-active-sessions` parameter in the `<pi_install_dir>/webgui/config/webgui.properties` file.

   The default value is 50.

4. Optional: To delete active sessions, enter the following command:

```
#  <pi_install_dir>/webgui/bin/cli.sh -u  <username>  -p  <password>  delete_sessions
```


Copyright © 2025 Ping Identity Corporation

> **Note**
>
> You need to have admin user privileges to delete active user sessions.

Result:

The current active users will be prompted to sign on again to the Dashboard.

Starting and stopping thePingIntelligenceDashboard

Start and stop the PingIntelligence Dashboard.

About this task

You can choose to start and stop all the components together or individually.

Steps

1. **To start and stop components:**

   Choose from:

   - **Start and stop components together.**
   - **It is recommended to start and stop components together using the following command:**

     ```
     # cd  <pi_install_dir>/pingidentity/webgui
     # ./bin/start-all.sh

     starting elasticsearch...
     warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
     warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
     OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and
     will likely be removed in a future release.
     elasticsearch started
     starting data-engine
     Data Engine configured for UTC timezone.
     PingIntelligence Data Engine 5.1 starting...
     data-engine started
     starting webgui
     WebGUI configured for UTC timezone.
     WebGUI 5.1 starting...
     please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for more details
     success: all ui components started
     ```

   - **To stop all components together, enter the following command:**

Copyright © 2025 Ping Identity Corporation

```
# cd  <pi_install_dir>/pingidentity/webgui
# ./bin/stop-all.sh

WebGUI 5.1
WebGUI is stopped.
PingIntelligence Data Engine 5.1
PingIntelligence Data Engine is stopped.
elasticsearch stopped
success: all ui components stopped
```

- Start and stop components individually.

    1. Start components in the following order:

        1. Start Elasticsearch by entering the following command:

            ```
            # cd  <pi_install_dir>/pingidentity/elasticsearch
            # ./bin/elasticsearch -d -p logs/elasticsearch.pid
            ```

            If Elasticsearch is running as a service, use the following command:

            ```
            # sudo systemctl start pi-elasticsearch.service
            ```

        2. Start the Dashboard by entering the following the command:

            ```
            # cd  <pi_install_dir>/pingidentity/webgui
            # ./bin/start.sh

            WebGUI configured for UTC timezone.
            WebGUI 5.1 starting...
            please see <pi_install_dir>/pingidentity/webgui/logs/admin/admin.log for
            more details
            ```

            If the Dashboard is running as a service, use the following command:

            ```
            # sudo systemctl start pi-dashboard.service
            ```

        3. Start the Web GUI by entering the following command:

            ```
            # cd  <pi_install_dir>/pingidentity/webgui
            # ./bin/start.sh
            ```

            If the Web GUI is running as a service, use the following command:

            ```
            # sudo systemctl start pi-webgui.service
            ```


Copyright © 2025 Ping Identity Corporation

2. Stop components in the following order:

    1. Stop the Web GUI by entering the following command:

    ```
    # cd  <pi_install_dir>/pingidentity/webgui
    # ./bin/stop.sh
    ```

    If Web GUI is running as a service, use the following command:

    ```
    # sudo systemctl stop pi-webgui.service
    ```

    2. Stop the Dashboard by entering the following command:

    ```
    # cd  <pi_install_dir>/pingidentity/webgui
    # ./bin/stop.sh

    WebGUI 5.1
    WebGUI is stopped.
    ```

    If the Dashboard is running as a service, use the following command:

    ```
    # sudo systemctl stop pi-dashboard.service
    ```

    3. Stop Elasticsearch by entering the following command:

    ```
    # cd  <pi_install_dir>/pingidentity/elasticsearch
    # kill -15 "$(<logs/elasticsearch.pid)"
    ```

    If Elasticsearch is running as a service, use the following command:

    ```
    # sudo systemctl stop pi-elasticsearch.service
    ```

Obfuscate keys and passwords

Using the PingIntelligence Dashboard command-line interface (CLI), you can obfuscate the keys and passwords configured in `dashboard.properties`.

The following keys and passwords are obfuscated:

- `abs.access_key`
- `abs.secret_key`
- `es.password`

Copyright © 2025 Ping Identity Corporation

The Dashboard ships with a default `dashboard_master.key`, which is used to obfuscate the keys and passwords. It is recommended to generate your own `dashboard_master.key`.

> **Note**
>
> During the process of obfuscation of keys and password, the Dashboard must be stopped. For more information, see Starting and stopping thePingIntelligenceDashboard.

The following diagram summarizes the obfuscation process:

Generate `dashboard_master.key`

You can generate the `dashboard_master.key` by running the `generate_obfkey` command in the Dashboard CLI:

```
/opt/pingidentity/dashboard/bin/cli.sh generate_obfkey -u admin -p
Password>

Please take a backup of config/dashboard_master.key before proceeding.

Warning: Once you create a new obfuscation master key, you should obfuscate all config keys also using
cli.sh obfuscate_keys

Warning: Obfuscation master key file /opt/pingidentity/dashboard/config/dashboard_master.key already exist.
This command will delete it create a new key in the same file

Do you want to proceed [y/n]: y

creating new obfuscation master key
Success: created new obfuscation master key at /opt/pingidentity/dashboard/config/dashboard_master.key
```

Obfuscate key and passwords

You can enter the keys and passwords in clear text in the Dashboard.properties file. You can run the `obfuscate_keys` command to obfuscate keys and passwords:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/dashboard/bin/cli.sh obfuscate_keys -u admin -p
Password>

Please take a backup of config/dashboard.properties before proceeding

Enter clear text keys and password before obfuscation.

Following keys will be obfuscated
 config/dashboard.properties: abs.access_key, abs.secret_key and es.password

Do you want to proceed [y/n]: y

obfuscating /opt/pingidentity/dashboard/config/dashboard.properties

Success: secret keys in /opt/pingidentity/dashboard/config/dashboard.properties obfuscated
```

You can start the Dashboard after passwords are obfuscated. For more information, see Starting and stopping thePingIntelligenceDashboard.

Important

After the keys and passwords are obfuscated and the Dashboard has started, move the `dashboard_master.key` to a secure location away from the Dashboard for security reasons. Before restarting the Dashboard, the `dashboard_master.key` must be present in the `` directory.

Configuring time zone in the PingIntelligence for APIs Dashboard

This topic discusses the steps involved in configuring the time zone settings in the PingIntelligence for APIs Dashboard.

About this task

You can set up the PingIntelligence Dashboard in either `local` or `utc` time zone. The Dashboard by default runs in the UTC time zone. You can configure the following parameters to set the time zone.

Parameter	File name	Description
`pi.webgui.server.timezone`	This parameter must be configured in `<pi_install_dir>/webgui/config/webgui.properties` file.	The valid values are `local` or `utc`. The default value is `utc`.
`dashboard.timezone`	This parameter must be configured in `<pi_install_dir>/dashboard/config/dashboard.properties` file.	The valid values are `local` or `utc`. The default value is `utc`.

The following is a snippet of `webgui.properties` for time zone parameter:

Copyright © 2025 Ping Identity Corporation

```
# Timezone configuration
# valid values: local, utc
pi.webgui.server.timezone=local

<truncated webgui.properties...>
```

The following is a snippet of `dashboard.properties` for time zone parameter:

```
# Timezone configuration
# valid values: local, utc
dashboard.timezone=local

<truncated dashboard.properties...>
```

> **Note**
>
> Make sure that the time zone parameter is set to the same value in both `webgui.properties` and `dashboard.properties`. Also make sure that ASE, the ABS AI engine, and the PingIntelligence for APIs Dashboard are all configured on the same time zone.

To change the time zone in PingIntelligence for APIs Dashboard, complete the following steps:

Steps

1. Stop the PingIntelligence for APIs Dashboard.
2. Update the `timezone` parameters in the `webgui.properties` and `dashboard.properties` files.
3. Start the PingIntelligence for APIs Dashboard.

> **Note**
>
> For more information, see Starting and stopping thePingIntelligenceDashboard.

PingIntelligence Dashboard properties

The Dashboard configuration file (`dashboard.properties`) is located in the `<pi_install_dir>/dashboard/config/` directory.

The following table explains the parameters and provides recommended values.

Parameter	Description
ABS IP, port, log level, and JKS password	
`abs.host`	ABS URL
`abs.port`	ABS port number. The default value is 8080.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`abs.ssl`	Set the value to enable or disable SSL connection with ABS. Valid values are `true` and `false`.
`abs.restricted_user_access`	Set the value to enable or disable restricted user access to ABS. Valid values are `true` and `false`.
`abs.access_key`	ABS access key.
`abs.secret_key`	ABS secret key.
`abs.query.interval`	ABS query polling interval in minutes. The default value is 10 minutes.
`abs.query.offset`	ABS query offset in minutes. The minimum value is 30 minutes.
Publish to UI, Elasticsearch distribution type, Elasticssearch URL, Elasticsearch username, Elasticsearch password, ILM policy, Kibana version	
`publish.ui.enable`	Set the value to enable or disable publishing of attack information and other metrics to the dashboard. Valid values are `true` and `false`.
`es.distro.type`	Elasticsearch distribution type. Valid values are `default` and `aws`.
`es.url`	Elasticsearch URL.
`es.username`	The username credential to Elasticsearch.
`es.password`	The password credentials to Elasticsearch.
`es.index.dashboard.activity.ilm.policy`	Location of Index Lifecycle Management (ILM) policy. If a policy is provided, it should be a valid JSON file. It is not a mandatory policy. The default directory is `<pi_install_dir>/dashboard/config/` directory.
`es.index.dashboard.activity.ism.policy`	Location of Index State Management (ISM) policy. If a policy is provided, it should be a valid JSON file and `es.distro.type` should be set to `aws`. It is not a mandatory policy. The default directory is `<pi_install_dir>/dashboard/config/` directory.
`kibana.version`	Kibana version. The default value is 6.8.1.
Log4j2 configuration properties	

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`publish.log4j2.enable`	Set the value to enable publishing attack details to Log4j2. Valid values `true` or `false`. By default, the Dashboard provides `syslog` support.
`log4j2.config`	Log4j2 configuration file to log attacks to an external service. For example, `Syslog`. Use `com.pingidentity.abs.publish` as the logger name in log4j2 configuration.
`log4j2.log.level`	Log4j2 log level for attack logging. The default value is `info`.
`log4j2.dependencies.dir`	The directory for any log4j2 config dependency jar's. This is useful for third-party log4j2 appenders. The default directory is `<pi_install_dir>/dashboard/plugins/`.
Log level, Time zone configuration	
`dashboard.log.level`	The applicable log levels. Valid values are `all`, `trace`, `debug`, `info`, `warn`, `error`, `fatal`, and `off`. The default value is `info`.
`dashboard.timezone`	Set time zone configuration for the Dashboard. Valid values are `local` or `utc`.
Dashboard fast forward properties **Note** The properties are only applicable if dashboard is started with the `start.sh --fast-forward` option.	
`dashboard.fastforward.earlier_time`	The Dashboard fast forward earlier time. The allowed format is `YYYY-MM-DDTHH:mm`.
`dashboard.fastforward.later_time`	The Dashboard fast forward later time. The allowed format is `YYYY-MM-DDTHH:mm`.
`dashboard.fastforward.query.range`	The Dashboard query range in minutes. It should be multiples of 10. The minimum value is 10.
`dashboard.fastforward.query.cooling_period`	Cooling period between each query polling batch in seconds. The minimum value is 30 seconds.

The following is a sample `dashboard.properties` file:

Copyright © 2025 Ping Identity Corporation

```
# Dashboard properties file

# ABS
# ABS Hostname/IPv4 address
abs.host=127.0.0.1
# ABS REST API port
abs.port=8080
# ABS SSL enabled ( true/false )
abs.ssl=true
# ABS Restricted user access ( true/false )
abs.restricted_user_access=false
# ABS access key
abs.access_key=OBF:AES:NuBmDdIhJM7KOB3BbXr4db5DfGJcrA==:hUsqFeTUmH5cOjiUPyws9WwTPYw9yAg0C1X1HSmSI30=
# ABS secret key
abs.secret_key=OBF:AES:NuBmDcAhXgsQu8qzJgIo1Mq97B/PVw==:7GpDn83ZAU6GRKYsZe86x0gdnYOZfTbi8rUimDW100o=
# ABS query polling interval (minutes)
abs.query.interval=10
# ABS query offset (minutes. minimum value 30 minutes)
abs.query.offset=30

# UI
# publish attacks+metrics to UI. Valid values true or false
publish.ui.enable=true
# elasticsearch Distribution Type
# valid values are default and aws
es.distro.type=default
# elasticsearch URL
es.url=https://localhost:9200/
# elasticsearch username. User should have manage_security privilege
# If elasticsearch is NOT configured with authentication security, leave this blank
es.username=elastic
# elasticsearch user password
es.password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=
# index lifecycle management (ILM) policy,it can be empty
# If a policy is provided, it should be a valid JSON file
es.index.dashboard.activity.ilm.policy=config/ilm.json
# index stae management (ISM) policy,it can be empty
# If a policy is provided, it should be a valid JSON file
es.index.dashboard.activity.ism.policy=config/ism.json
# kibana version
kibana.version=6.8.1

# Log4j2
# publish attacks to Log4j2. Valid values true or false
# By default it provides syslog support
publish.log4j2.enable=false
# log4j2 config file to log attacks to an external service. For example, Syslog
# use com.pingidentity.abs.publish as logger name in log4j2 configuration
log4j2.config=config/syslog.xml
# log4j2 log level for attack logging
log4j2.log.level=INFO
# directory for any log4j2 config dependency jar's.
# useful for third party log4j2 appenders
# it should be a directory
```

Copyright © 2025 Ping Identity Corporation

```
log4j2.dependencies.dir=plugins/

# Log level
dashboard.log.level=INFO

# Timezone configuration
# valid values: local, utc
dashboard.timezone=local

## Fastforward. Only applicable if dashboard is started with 'start.sh --fast-forward'

# earlier time. format YYYY-MM-DDTHH:mm
# E.g 2019-07-12T10:00
dashboard.fastforward.earlier_time=2019-07-12T10:00

# later time. format YYYY-MM-DDTHH:mm
# E.g 2019-11-13T23:50
dashboard.fastforward.later_time=2019-11-13T23:50

# query range in minutes. It should be multiple of 10
# minimum value is 10
dashboard.fastforward.query.range=60

# cooling period between each query polling batch in seconds
# minimum value 30 seconds
dashboard.fastforward.query.cooling_period=60
```

PingIntelligence Dashboard WebGUI properties

The WebGUI configuration file (`webgui.properties`) is located in the `<pi_install_dir>/webgui/config/` directory.

The following table explains the parameters and provides recommended values.

Parameters	Description
Server, time zone properties	
`pi.webgui.server.port`	WebGUI sever port number. The default value is 8030. **Note** You can specify the port number as 443 to run WebGUI on HTTPS. This option is only available if WebGUI `start.sh` is executed by a root user.
`pi.webgui.server.timezone`	The time zone configuration for WebGUI. Valid values are `local` or `utc`. The default value is `utc`.
Log level, authentication mode properties	
`pi.webgui.admin.log.level`	The applicable log levels. Valid values are `all`, `trace`, `debug`, `info`, `warn`, `error`, `fatal`, and `off`. The values are not case sensitive.

Copyright © 2025 Ping Identity Corporation

Parameters	Description
`pi.webgui.server.authentication-mode`	The authentication mode. Valid values are `native` or `sso`.
Session properties	
`pi.webgui.session.max-age`	The maximum allowed duration for a session. After max-age duration, the user will be asked to re-authenticate. The allowed format is `<duration number>m` (minutes), `<duration number>h` (hours), or `<duration number>d` (days). For example, 20m, 20h, or 20d. **Note** The duration value must be greater than zero.
`pi.webgui.session.expiry-time`	The maximum duration allowed for a session to remain inactive. The value should be provided in minutes. After an inactivity period, the user will be asked to re-authenticate.
`pi.webgui.session.max-active-sessions`	The maximum number of active sessions allowed. The default value is 50.
SSL properties	
`pi.webgui.server.ssl.enabled-protocols`	The supported SSL enabled protocols. For more information, see .oracle.com/en/java/javase/11/docs/specs/security/standard-names.html//[]. For multiple SSL protocols, use a comma-separated list. For example, TLSv1.1, TLSv1.2.
`pi.webgui.server.ssl.ciphers`	The supported SSL ciphers. For the list of valid cipher names, see .oracle.com/en/java/javase/11/docs/specs/security/standard-names.html//[]. For multiple cipher names, use a comma-separated list. For example, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.
`pi.webgui.server.ssl.key-store`	The SSL server key store location value. For JKS key store, the key store password and private key password should be same when you generate a JKS key store.
`pi.webgui.server.ssl.key-store-type`	The SSL key store type. The default value is `jks`.
`pi.webgui.server.ssl.key-store-password`	The password credentials to the SSL key store.
`pi.webgui.server.ssl.key-alias`	Alias for the SSL key. The default value is `PingIntelligence`.
ABS properties	
`pi.webgui.abs.url`	ABS URL.

Copyright © 2025 Ping Identity Corporation

Parameters	Description
`pi.webgui.abs.access-key`	ABS access key.
`pi.webgui.abs.secret-key`	ABS secret key.
`pi.webgui.abs.api-service-url`	Host URL for the API Publish Service. The default port number is 8020.
ASE properties	
`pi.webgui.ase.url`	ASE management URL value. **Note** The ASE management URL is an optional parameter.
`pi.webgui.ase.mode`	ASE deployment mode. Valid values are `inline` or `sideband`. When PingIntelligence is deployed on cloud, the default value is `inline`.
`pi.webgui.ase.access-key`	ASE access key.
`pi.webgui.ase.secret-key`	ASE secret key.
Kibana properties	
`pi.webgui.dashboard.url`	The Kibana URL.
`pi.webgui.dashboard.username`	The Kibana username credentials.
`pi.webgui.dashboard.password`	The Kibana password credentials.
Elasticsearch properties	
`pi.webgui.elasticsearch.url`	Elasticsearch URL.
`pi.webgui.elasticsearch.username`	The username credential to Elasticsearch.
`pi.webgui.elasticsearch.password`	The password credentials to Elasticsearch.
`pi.webgui.elasticsearch.distro-type`	Elasticsearch distribution type. Valid values are `default` and `aws`.
API discovery properties	
`pi.webgui.discovery.source`	Source for API discovery. Valid values are `abs`, `axway`, and `pingaccess`.
Indicators of Attack (IoA) listing properties	

Copyright © 2025 Ping Identity Corporation

Parameters	Description
`pi.webgui.ioclisting.fetchsize`	**The limit of documents that can be pulled from Elasticsearch. The default value is 2000. The upper limit is 10,000.**
H2 database properties	
`pi.webgui.datasource.url`	**H2 database URL. The database is started on the default port number 9092. The total number of documents that can be fetched in an Elasticsearch search query to list IoAs for different client identifier types.**
`pi.webgui.datasource.username`	**Username credentials to the H2 database.**
`pi.webgui.datasource.password`	**Password to the H2 database.**
`pi.webgui.datasource.encryption-password`	**Password to encrypt the H2 database.**

> **Note**
> The H2 database will use the properties when it is first started. If you want to change them, you can stop the WebGUI server and delete the `data/h2` directory and start again. When you delete the `data/h2` directory, WebGUI is reset. The login passwords, login sessions, and API state information is lost when the WebGUI is reset.

Connection timeout properties	
`pi.webgui.http-client.timeout`	**Total number of documents that can be fetched in an Elasticsearch TCP connection timeout value in milliseconds. Timeout after which TCP connection to ABS, ASE, Dashboard, Elasticsearch is closed by the WebGUI.**
`pi.webgui.http-client.socket-timeout`	**Socket timeout value in milliseconds. Timeout after which socket to ABS, ASE, Dashboard, Elasticsearch is closed by the WebGUI.**
JDK truststore properties	
`pi.webgui.jdk.truststore`	**The location of JDK truststore. The default value is `$JAVA_HOME/lib/security/cacerts`.**
`pi.webgui.jdk.truststore-password`	**The password to JDK truststore.**

> **Note**
> You can configure the values of JDK trustore and its password only if the defaults don't match.

HTTP client connection properties	
`pi.webgui.http-client.max-connections`	**Maximum allowed HTTP connections.**

Copyright © 2025 Ping Identity Corporation

Parameters	Description
`pi.webgui.http-client.request-timeout`	**Request timeout for the HTTP clients.**
`pi.webgui.http-client.keep-alive-time`	**Connection keep-alive time.**
`pi.webgui.http-client.idle-time`	**HTTP client idle time.**

A sample `webgui.properties` file is displayed below:

Copyright © 2025 Ping Identity Corporation

```
# PingIntelligence WebGUI properties file
# This is in standard java properties file format
# comments are denoted by number sign (#) as the first non blank character
# multiline values are ended with '\' as end of line

# server listening port
# server listens on 0.0.0.0 ( all interfaces )
# server enables only https(ssl) on this port
pi.webgui.server.port=8030

# Timezone configuration
# valid values: local, utc
pi.webgui.server.timezone=utc

# log level
# valid values: ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF
# filtering sequence: ALL > TRACE > DEBUG > INFO > WARN > ERROR > FATAL > OFF
# higher level in the sequence will allow all the lower level log messages
# case insensitive
pi.webgui.admin.log.level=INFO

# Authentication mode
# valid values: native, sso
pi.webgui.server.authentication-mode=native

# ui login session
# maximum duration of a session
# after max-age duration, user will be asked to re-authenticate
# format: <duration>m (minutes) /h (hours) /d (days)
# duration should be > 5 minutes
pi.webgui.session.max-age=6h

# maximum session inactivity duration( No requests from the session ). In minutes
# after inactivity period, user will be asked to re-authenticate
pi.webgui.session.expiry-time=30

# maximum active sessions allowed
pi.webgui.session.max-active-sessions=50

 server ssl properties
# ssl enabled protocols ( https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-
names.html#sslcontext-algorithms)
# for multiple SSL protocols use comma separated list. e.g TLSv1.1,TLSv1.2
pi.webgui.server.ssl.enabled-protocols=TLSv1.2

# supported ssl ciphers
# valid cipher names: https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-
names.html#jsse-cipher-suite-names
# for multiple cipher names use comma separated list. e.g
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
pi.webgui.server.ssl.ciphers=TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,TLS_DHE_RSA_WITH
\
TLS_RSA_WITH_AES_256_CBC_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECD
\
```

Copyright © 2025 Ping Identity Corporation

```
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDH
\
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_DHE_RS
\
TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES
\
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,TLS_
\
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECD

 server ssl keystore
# for JKS keystore, keystore password and private key password should be same when you generate a jks
keystore.
pi.webgui.server.ssl.key-store=config/webgui.jks
pi.webgui.server.ssl.key-store-type=JKS
pi.webgui.server.ssl.key-store-password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:
+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=
pi.webgui.server.ssl.key-alias=PingIntelligence

# abs properties
pi.webgui.abs.url=https://localhost:8080
pi.webgui.abs.access-
key=OBF:AES:NuBmDdIhJM7KOB3BbXr4db5DfGJcrA==:hUsqFeTUmH5cOjiUPyws9WwTPYw9yAg0C1X1HSmSI30=
pi.webgui.abs.secret-key=OBF:AES:NuBmDcAhXgsQu8qzJgIo1Mq97B/PVw==:
7GpDn83ZAU6GRKYsZe86x0gdnYOZfTbi8rUimDW100o=

# ase properties
# ASE management url
pi.webgui.ase.url=https://localhost:8010
# ASE mode: valid values: inline or sideband
pi.webgui.ase.mode=inline
pi.webgui.ase.access-key=OBF:AES:NuZ4O93cWBKyKDFOZFINHeBew8sQ:eu//E2CIObNNGvFOfHrLuAuec4WvN4yZsThAea4iBLA=
pi.webgui.ase.secret-key=OBF:AES:NuZ4O93cWBKyKDFOZFINHeBew8sQ:eu//E2CIObNNGvFOfHrLuAuec4WvN4yZsThAea4iBLA=

# kibana rendering ( dashboard ) properties
pi.webgui.dashboard.url=https://localhost:5601
pi.webgui.dashboard.username=ping_user
pi.webgui.dashboard.password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:
+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=

# elasticsearch properties
pi.webgui.elasticsearch.url=https://localhost:9200
pi.webgui.elasticsearch.username=elastic
pi.webgui.elasticsearch.password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:
+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=
# ES distribution type
# valid values: default, aws
pi.webgui.elasticsearch.distro-type=default

# api discovery properties
# discovery source
# valid values: abs, axway and pingaccess
# for axway and pingaccess, see config/discovery.properties
pi.webgui.discovery.source=abs
```


Copyright © 2025 Ping Identity Corporation

```
# ioc listing properties
# total number of documents that can be fetched in an elasticsearch search
# query to list iocs for different client identifier types.
pi.webgui.ioclisting.fetchsize=2000

# server internal configurations

 local h2 db datasource properties
# h2 db is started on default port 9092
pi.webgui.datasource.url=jdbc:h2:ssl://localhost/webgui_data;CIPHER=AES

# h2 db will use following properties when it is first started. There is no way to change it afterwards
# If you want to change it, you should stop webgui server and delete data/h2 directory and start again.
# when you delete data/h2 directory, webgui is reset. you will loose login passwords/login sessions/api
state info.
pi.webgui.datasource.username=sa
pi.webgui.datasource.password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:
+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=
pi.webgui.datasource.encryption-password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:
+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=

# server to abs/ase/dashboard http connection properties
# tcp connect timeout in milliseconds
pi.webgui.http-client.timeout=15000
# timeout after which socket to abs/ase/dashboard/elasticsearch is closed by the webgui
pi.webgui.http-client.socket-timeout=120000

## http client connection pool configurations
pi.webgui.http-client.max-connections=256
pi.webgui.http-client.request-timeout=30000
pi.webgui.http-client.keep-alive-time=120000
pi.webgui.http-client.idle-time=120000
```

PingIntelligence Dashboard engine

When you install the PingIntelligence Dashboard, the on-prompt installation steps asks for configuration values, including access and secret key, ABS and ASE URL, and so on.

These values after installation are populated in the `<pi_install_dir>/dashboard/config/dashboard.properties` file. To change these values, you can stop the Dashboard engine, edit the `dashboard.properties` file and then start the Dashboard engine. See Starting and stopping thePingIntelligenceDashboard for more information on how to start and stop each component individually.

Copyright © 2025 Ping Identity Corporation

```
# Dashboard properties file

# ABS
# ABS Hostname/IPv4 address
abs.host=127.0.0.1
# ABS REST API port
abs.port=8080
# ABS SSL enabled ( true/false )
abs.ssl=true
# ABS Restricted user access ( true/false )
abs.restricted_user_access=true
# ABS access key
abs.access_key=OBF:AES:NuBmDdIhQeNlRtU8SMKMoLaSpJviT4kArw==:HHuA9sAPDiOen3VU+qp6kMrkgNjAwnKO6aa8pMuZkQw=
# ABS secret key
abs.secret_key=OBF:AES:NuBmDcAhQeNlPBDmyxX+685CBe8c3/STVA==:BIfH+FKmL5cNa1DrfVuyc5hIYjimqh7Rnf3bv9hW0+4=
# ABS query polling interval (minutes)
abs.query.interval=10
# ABS query offset (minutes. minimum value 30 minutes)
abs.query.offset=30

# UI
# publish attacks+metrics to UI. Valid values true or false
publish.ui.enable=true
# elasticsearch URL
es.url=https://localhost:9200/
# elasticsearch username. User should have manage_security privilege
es.username=elastic
# elasticsearch user password
es.password=OBF:AES:NOp0PNQvc/RLUN5rbvZLtTPghqVZzD9V:+ZGHbhpY4HENYYqJ4wn50AmoO6CZ3OcfjqTYQCfgBgc=
# kibana version
kibana.version=6.8.1

# Log4j2
# publish attacks to Log4j2. Valid values true or false
# By default it provides syslog support
publish.log4j2.enable=false
# log4j2 config file to log attacks to an external service. For example, Syslog
# use com.pingidentity.abs.publish as logger name in log4j2 configuration
log4j2.config=config/syslog.xml
# log4j2 log level for attack logging
log4j2.log.level=INFO
# directory for any log4j2 config dependency jar's.
# useful for third party log4j2 appenders
# it should be a directory
log4j2.dependencies.dir=plugins/

# Log level
dashboard.log.level=INFO
```

The following table describes all the parameters in the `dashboard.properties` file.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
ABS	
`abs.host`	**IP address of the ABS server.** **Note** Two options exist to choose an ABS server: • Utilize an existing ABS server. • For production deployments, Ping Identity recommends dedicating an exclusive ABS reporting node.
`abs.port`	**REST API port number of the ABS host. See `abs.properties`. The default value is 8080.**
`abs.ssl`	**Setting the value to true ensures SSL communication between ABS and dashboard engine.**
`abs.restricted_user`	**When set to `true`, Elasticsearch uses the restricted user header (configured in the `pingidentity/abs/mongo/abs_init.js` file) to fetch the obfuscated values of OAuth token, cookie, and API keys. When set to `false`, the admin user header is used to fetch the data in plain text. For more information on admin and restricted user headers, see ABS users for API reports.**
`abs.access_key`	**Access key from ABS. See `pingidentity/abs/mongo/abs_init.js`. Make sure to enter the access key based on the value set in the previous variable. For example, if `abs.restricted_user` is set to `true`, then enter the access key for restricted user. If `abs.restricted_user` is set to `false`, then use the access key for the admin user.**
`abs.secret_key`	**Secret key from ABS. See `pingidentity/abs/mongo/abs_init.js`. Make sure to enter the secret key based on the value set in the previous variable. For example, if `abs.restricted_user` is set to `true`, then enter the secret key for restricted user. If `abs.restricted_user` is set to `false`, then use the secret key for the admin user.**
`abs.query.interval`	**Polling interval to fetch data from ABS. The default is 10 minutes.**
`abs.query.offset`	**The time required by ABS to process access logs and generate result. The minimum and default value is 30 minutes.**
UI	
`publish.ui.enable`	**Set to `true` to display the PingIntelligence Dashboard. The Dashboard displays attack and metrics data. Set to `false` if you do not want to display the Dashboard.**
`es.url`	**Elasticsearch URL.**

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`es.username`	Elasticsearch username.
`es.password`	Elasticsearch password.
`kibana.version`	Kibana version. The default is 6.8.1.
`dashboard.log.level`	Log level for the Dashboard. The default log level is `INFO`. Another log level is `DEBUG`.
Log4j	
`publish.log4j2.enable`	Set to `true` to send attack data to the `syslog` server. Set to `false` to disable sending attack data to syslog server. **Note** The Dashboard and `Syslog` cannot be disabled together.
`log4j2.config`	The log4j2 config file that logs the attack data.
`log4j2.log.level`	Log level for log4j. The default log level is `INFO`.
`log4j2.dependencies.dir`	The directory for any log4j configuration dependency. Make sure that it is a directory.

Dashboard engine fast forward

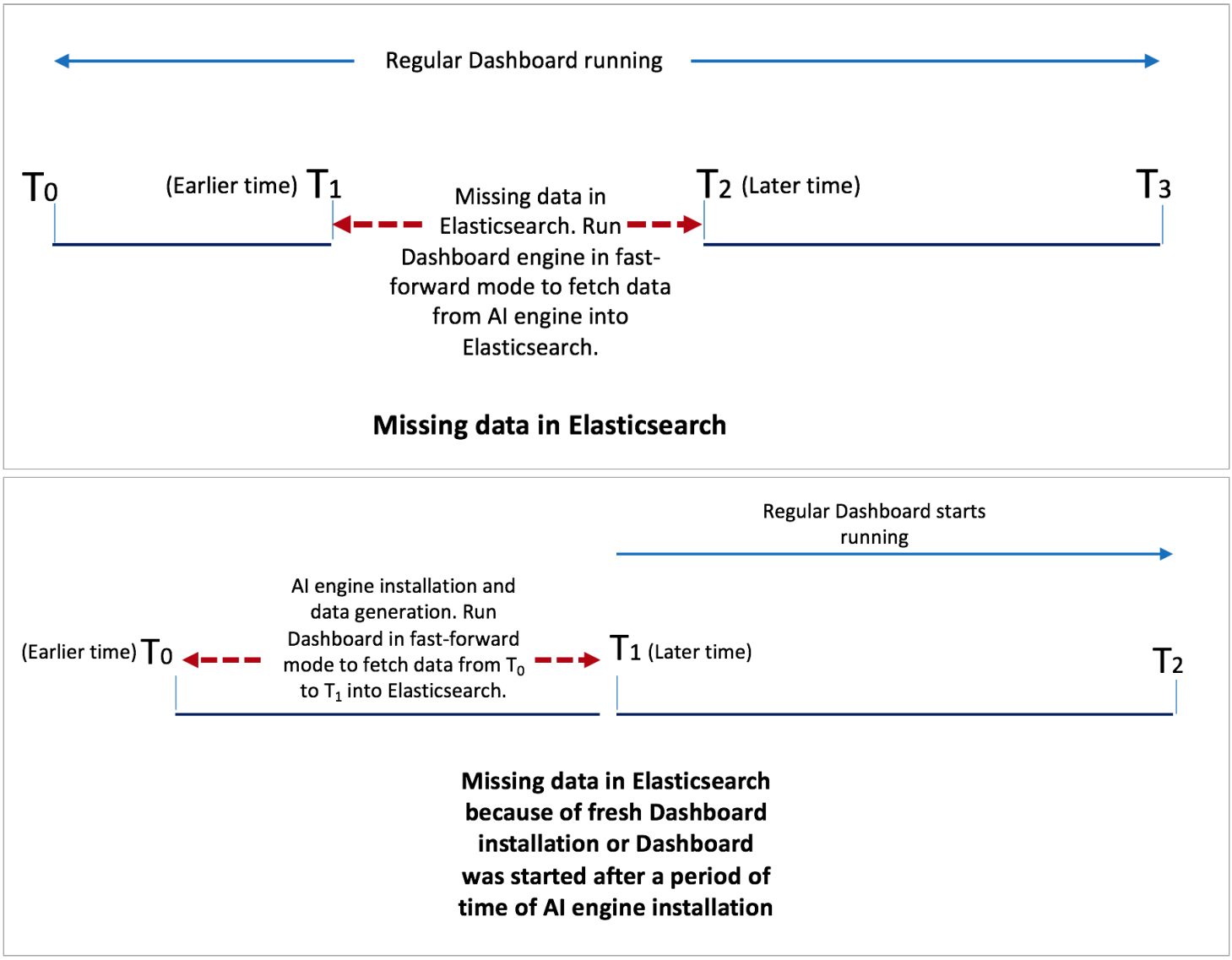
Start PingIntelligence Dashboard in fast-forward mode to populate the Dashboard with historical data.

Possible scenarios in which running the Dashboard in fast-forward mode is useful are:

- Elasticsearch data was accidentally deleted, and you want to repopulate the Dashboard.
- The Dashboard was not available for a specific duration of time, and you wish to fetch the data for that time duration.
- The Dashboard was installed after the other PingIntelligence components were deployed, and you want to populate the Dashboard with data from when PingIntelligence was first started.

The following diagrams summarize the use case for Dashboard's fast-forward mode:

Copyright © 2025 Ping Identity Corporation



When you run the Dashboard in fast-forward mode, it fetches data from a time frame you define in `YYYY-MM-DDTHH:mm` format in the `dashboard.properties` file. For example, if you want to fetch data from January 1, 2019 01:00 to March 31, 2019 23:00, then `earlier-date` in `dashboard.properties` would be 2019-01-01T01:00 and `later-date` would be 2019-03-31T23:00.

The Dashboard stops querying the AI engine when its query reaches the later date. The Dashboard stopping time is logged in the `logs/dashboard_fastforward.log` file along with the other Dashboard activities. The `logs/dashboard_fastforward.log` file is rotated every 24 hours. You can see the data visualization of the specified period in the Dashboard UI already running.

> **Note**
>
> If your current Dashboard engine is running in `/opt/pingidentity/dashboard/`, make sure that you use a different directory to run the Dashboard in fast-forward mode, for example: `/opt/pingidentity/dashboard_fast_forward/`.

Copy the Dashboard binary and configure the `dashboard.properties` file with `earlier-date` and `later-date` in the `Fastforward` section of the properties file. The following table shows the available parameters for the Dashboard fast-forward mode.

Copyright © 2025 Ping Identity Corporation

Parameter	Description
`dashboard.fastforward.earlier_time`	The query start date and time in `YYYY-DD-MMTHH:mm` format.
`dashboard.fastforward.later_time`	The query end date and time in `YYYY-DD-MMTHH:mm` format.
`dashboard.fastforward.query.range`	The time in minutes that the Dashboard queries the AI engine in a single pass.
`dashboard.fastforward.query.cooling_period`	The time in seconds between two Dashboard queries to the AI engine. The minimum and the default value is 60 seconds.

The following is an example of the `Fastforward` section of the `dashboard.properties` file.

```
## Fastforward. Only applicable if dashboard is started with 'start.sh --fast-forward'

# earlier time. format YYYY-MM-DDTHH:mm
# E.g 2019-07-12T10:00
dashboard.fastforward.earlier_time=2019-07-12T10:00

# later time. format YYYY-MM-DDTHH:mm
# E.g 2019-11-13T23:50
dashboard.fastforward.later_time=2019-11-13T23:50

# query range in minutes. It should be multiple of 10
# minimum value is 10
dashboard.fastforward.query.range=60

# cooling period between each query polling batch in seconds
dashboard.fastforward.query.cooling_period=60
```

Start dashboard engine in fast-forward mode

You can install a new instance of dashboard binary in a different directory in `/opt/pingidentity/`, for example: `/opt/pingidentity/dashboard_fast_forward`. You can enter the following command to start the Dashboard in fast-forward mode:

```
# /opt/pingidentity/dashboard_fast_forward/bin/start.sh --fast-forward
starting Dashboard Fastforward 4.1
```

syslog">

Configuring the PingIntelligence Dashboard engine for `syslog`

The PingIntelligence Dashboard engine supports sending attack information to a `syslog` server.

About this task

Copyright © 2025 Ping Identity Corporation

The PingIntelligence Dashboard ships with a `syslog.xml` and `attack_log.xml` file in the Dashboard `config` directory. The `config` file supports other formats available with Log4j including `.properties`, `.json`, or `.yml`.

The following is a snippet of the `syslog.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="APIIntelligence" packages="">
  <Appenders>
    <!--<Syslog name="bsd" host="localhost" port="514" protocol="TCP"
      ignoreExceptions="false" immediateFlush="true" />-->
    <Syslog name="RFC5424"  host="localhost" port="614" protocol="TCP"
      format="RFC5424"  appName="APIIntelligence" mdcId="mdc"
      facility="LOCAL0" enterpriseNumber="18060" newLine="true"
      messageId="Audit" id="App" ignoreExceptions="false" immediateFlush="true"/>
  </Appenders>
  <Loggers>
     <Logger name="com.pingidentity.abs.publish" level="info" additivity="false">
      <AppenderRef ref="RFC5424"/>
    </Logger>
  </Loggers>
</Configuration>
```

Steps

1. Enable `syslog` support by editing the `dashboard.properties` file.

   By default `syslog` is disabled. The Dashboard uses Log4j version 2.11.2 to publish attack data to `syslog`.

   The following is a snippet of `dashboard.properties` with `syslog` enabled.

   ```
   # Log4j2
   # publish attacks to Log4j2. Valid values true or false
   # By default it provides syslog support
    publish.log4j2.enable=true
   # log4j2 config file to log attacks to an external service. For example, Syslog
   # use com.pingidentity.abs.publish as logger name in log4j2 configuration
    log4j2.config=config/syslog.xml
   # log4j2 log level for attack logging
   log4j2.log.level=INFO
   # directory for any log4j2 config dependency jar's.
   # useful for third party log4j2 appenders
   # it should be a directory
   log4j2.dependencies.dir=plugins/
   ```

   Result:

   The attack data is published to a Log4j logger named `com.pingidentity.abs.publish`. The Log4j configuration file must have a logger named `com.pingidentity.abs.publish`. Any Log4j2 config file that wants to capture attack data from Dashboard must have at least one logger with name `com.pingidentity.abs.publish`.

2. Configure the server and port number of the `syslog` server in `config/syslog.xml` file.

   The following is a snippet of the `syslog.xml` file displaying the server and port number parameters:

Copyright © 2025 Ping Identity Corporation

```
<!--  # Syslog RFC5424 format, TCP -->
   <Syslog name="TCP_RFC5424"
      host="localhost" port="614"
     appName="APIIntelligence"
     id="App"
     enterpriseNumber="18060"
     facility="LOCAL0"
     messageId="Audit"
     format="RFC5424"
     newLine="true"
     protocol="TCP"
     ignoreExceptions="false"
     mdcId="mdc" immediateFail="false" immediateFlush="true"
     connectTimeoutMillis="30000" reconnectionDelayMillis="5000"/>
```

Configure authentication - SSO with PingFederate

PingIntelligence for APIs Dashboard provides two methods for user authentication: native or single sign-on (SSO).

You can configure the authentication method by configuring `pi.webgui.server.authentication-mode` property in the `<pi_install_dir>/pingidentity/webgui/config/webgui.properties` file. The default authentication method is `native`.

> **Note**
>
> SSO authentication should be used only for production deployments. Use native authentication for Proof of Concept deployments.

SSO configuration for PingIntelligence for APIs Dashboard

SSO configuration for the PingIntelligence Dashboard involves configuring both Dashboard and PingFederate. The following is a summary of configuration steps:

1. Verify the prerequisites.
2. Configure an OAuth client in PingFederate.
3. Configure the `webgui.properties` file.
4. Configure the `sso.properties` file in the Dashboard.
5. Import the PingFederate SSL server certificate.
6. Obfuscate `sso.properties`.
7. Start the Dashboard.

Verify the prerequisites

Ensure the following prerequisites are complete before SSO configuration:

- PingFederate is installed and configured to support OpenID Connect (OIDC) SSO for any client. The current supported PingFederate versions are 9.3 or 10.1.

Copyright © 2025 Ping Identity Corporation

- PingIntelligence for APIs Dashboard is installed.

Configure OAuth client in PingFederate

Creating and configuring an OAuth client in PingFederate is an essential step for PingIntelligence Dashboard's SSO authentication. If the OAuth client is not correctly configured in PingFederate, authentication failure will occur. To configure an OAuth client, complete the steps in Configuring an OAuth client in PingFederate for PingIntelligence Dashboard SSO.

Configure `webgui.properties` file

Edit the `<pi_install_dir>/pingidentity/webgui/config/webgui.properties` to set the value of `pi.webgui.server.authentication-mode` to `sso` to configure authentication using SSO.

```
# Authentication mode
# valid values: native, sso
 pi.webgui.server.authentication-mode=sso
```

Configure SSO properties file in Dashboard

Configure the `<pi_install_dir>/pingidentity/webgui/sso.properties` file to complete the PingIntelligence Dashboard's SSO authentication. For more information, see Configuring Dashboard `sso.properties` for PingFederate.

Import the PingFederate SSL server certificate

After the PingIntelligence Dashboard configuration for SSO is complete, import PingFederate's SSL server certificate to the PingIntelligence Dashboard's truststore `<pi_install_dir>/pingidentity/webgui/config/webgui.jks`.

Complete the following steps to import SSL certificate:

1. Copy PingFederate's SSL server certificate to `<pi_install_dir>/pingidentity/webgui/config/ directory`.
2. Execute the following command:

```
# cd <pi_install_dir>/pingidentity/webgui/config/
keytool -import -trustcacerts -file <pf_certificate.crt> -alias pi-sso -keystore webgui.jks
```

> **Note**
>
> The default password to import `pf_certificate.crt` to `webgui.jks` is `changeme`.

Obfuscate `sso.properties`

You can obfuscate keys added in SSO properties using the following commands:

```
# cd <pi_install_dir>/pingidentity/webgui
# ./bin/cli.sh obfuscate_keys
```

Start PingIntelligence for APIs Dashboard

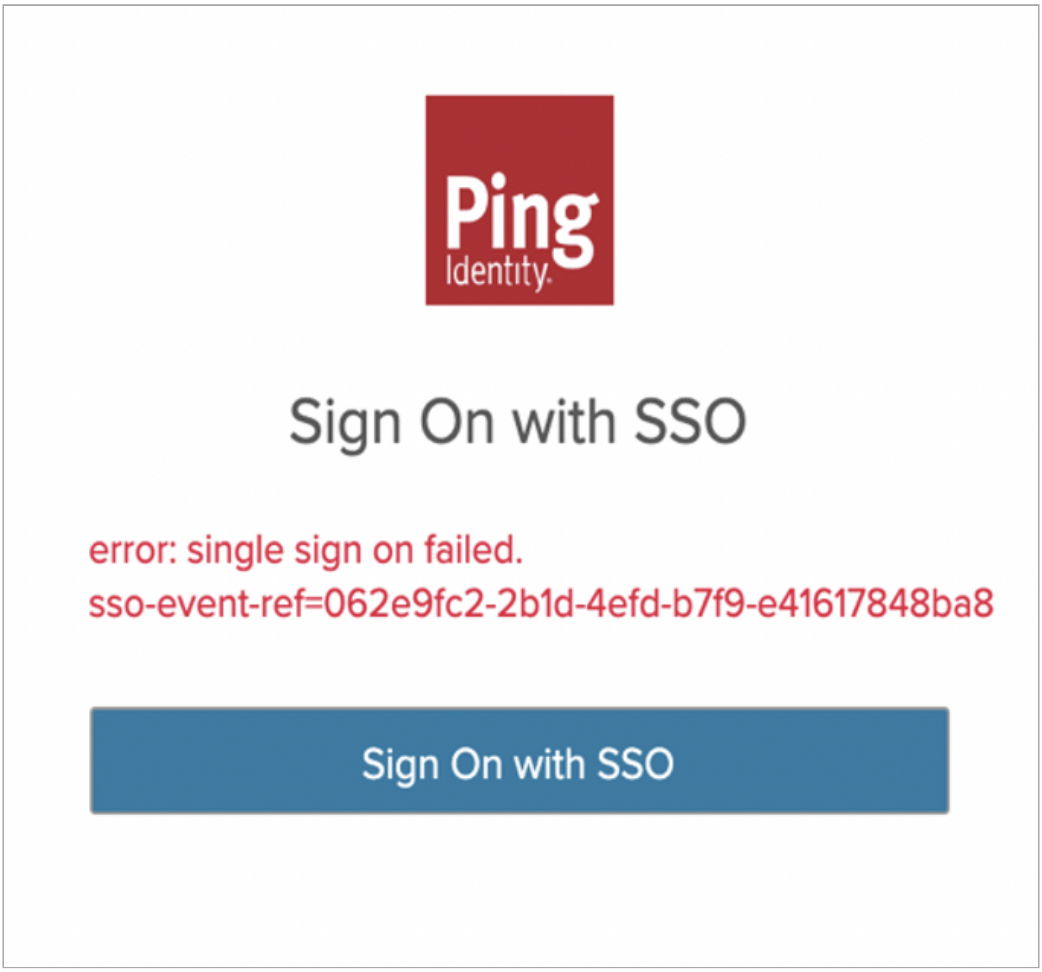
Start the PingIntelligence for APIs Dashboard. For more information, see Starting and stopping thePingIntelligenceDashboard.

Copyright © 2025 Ping Identity Corporation

When the PingIntelligence Dashboard is started successfully, access it using `https://<pi_install_host>:8030`. The Dashboard will start SSO Authentication, and a new session will get created for the logged-in users.

> **Note**
>
> Every PingIntelligence Dashboard SSO authentication event is attached with a `unique ID`, which is logged in `<pi_install_dir>/pingidentity/webgui/logs/admin/sso.log`.

If SSO authentication fails for any reason, PingIntelligence Dashboard shows the following error message.



> **Note**
>
> You can filter `sso-event-ref = <unique ID>` in the `<pi_install_dir>/pingidentity/webgui/logs/admin/sso.log` file to find the reason for SSO failure.

Configuring an OAuth client in PingFederate for PingIntelligence Dashboard SSO

Configure an OAuth client in PingFederate for PingIntelligence Dashboard single sign-on (SSO).

About this task

For more information on creating and configuring an OAuth client in PingFederate, see Managing OAuth clients.

Copyright © 2025 Ping Identity Corporation

Steps

- **Create and configure an OAuth client in PingFederate with the following configuration details.**

Option	Description
Client ID	**Create an OAuth client in PingFederate with Client ID as PingIntelligence. You can use any other value for Client ID in place of PingIntelligence.**
Client Authentication	**The current release of the PingIntelligence Dashboard supports NONE and CLIENT SECRET authentication methods.** **Client TLS Certificate authentication and Private Key JWT based authentication are not supported by the Dashboard.** **When CLIENT SECRET is selected as the client authentication method, you can generate a random client secret or use a custom secret, which is used by the PingIntelligence Dashboard for client authentication.**
Require Signed Request	**Do not enable.** **Important** The PingIntelligence Dashboard does not support signed requests.
Redirection URIs	**Set the redirection URI in the PingFederate OAuth client configuration. The path in the URI is as follows: https://*pi_install_host*:8030/login/oauth2/code/PingIntelligence. Do not change the path in the URI, just substitute the hostname. For example,** https://172.16.40.180:8030/login/oauth2/code/PingIntelligence**.**

Copyright © 2025 Ping Identity Corporation

Option	Description
Claims	**The following Claims must be configured in PingFederate and are mandatory for a successful authentication of a logged in user in the PingIntelligence Dashboard.** ◦ **A Claim for Subject Identifier, which should provide the unique identifier for the logged in user.** ◦ **A Claim for providing First Name.** ◦ **A Claim for providing Last Name.** ◦ **A Claim for providing the Role information.** **Note** The PingIntelligence Dashboard fetches the claims for an authenticated User from the PingFederate UserInfo endpoint. In PingIntelligence 4.4, the supported values for the Role Claim are ADMIN and REGULAR. They are case-sensitive; if a blank or any other value is configured, SSO will fail. Roles assigned to Users with in an enterprise should be mapped to ADMIN or REGULAR. PingIntelligence 4.4.1 and later versions support both single or multiple values for the Role Claim. If you are configuring the Role Claim with a single value then the allowed values are ADMIN and REGULAR and they are case-sensitive. If multiple values are sent, then one of the values must end with either of the following, and the values are not case-sensitive: ◦ *Ping-Dashboard-Admin* ◦ *Ping-Dashboard-Regular* If multiple values are configured for the Role Claim and one of them is an Admin role, then the Admin role takes a precedence.
Scopes	**The Scopes required to be configured in PingFederate for the PingIntelligence Dashboard application are:** ◦ **Mandatory Scopes:** `profile` **and** `openid` ◦ **Additional Scopes** **Note** The Claims configured for the PingIntelligence Dashboard can be mapped to the Mandatory Scope profile or to one or more Additional Scopes.

Copyright © 2025 Ping Identity Corporation

Option	Description
Allowed Grant Types	Enable Authorization Code. The PingIntelligence Dashboard supports only Authorization Code as the grant type.
Restrict Response Types	If enabled, select code.
Proof Key For Code Exchange (PKCE)	Do not enable. **Important** The PingIntelligence Dashboard does not support PKCE.
ID Token Signing Algorithm	The supported ID token signing algorithms are: ◦ Default ◦ RSA using SHA-256
ID Token Key Management Encryption Algorithm	Select No Encryption because encryption is not supported by the PingIntelligence Dashboard.

sso.properties for PingFederate">

Configuring Dashboard `sso.properties` for PingFederate

To complete the Dashboard's single sign-on (SSO) authentication, configure the `<installation_path>/pingidentity/webgui/sso.properties` file.

The following table describes the SSO properties.

Property	Description
`pi.webgui.sso.oidc.provider.issuer-uri`	Configure the URI of the OpenID Connect (OIDC) service provider (PingFederate). For example, `pi.webgui.sso.oidc.provider.issuer-uri=https://pf_installed_host:9031.` The Dashboard retrieves the PingFederate OpenID Provider configuration using the following URL: `<pi.webgui.sso.oidc.provider.issuer-uri>/.well-known/openid-configuration`. **Note** This is a mandatory property.

Copyright © 2025 Ping Identity Corporation

Property	Description
`pi.webgui.sso.oidc.client.id`	**Configure the OIDC client ID. The default value is `PingIntelligence`. Make sure to configure the same value in PingFederate. If you want to change the default value, change the client ID in PingFederate as well. For more information, see Configuring an OAuth client in PingFederate for PingIntelligence Dashboard SSO.** **Note** This is a mandatory property.
`pi.webgui.sso.oidc.client.secret`	**Set the client secret value in plain-text of the OAuth client created for the Dashboard application in PingFederate. The secret value is obfuscated in the Dashboard. The default value configured in `sso.properties` is `changeme`.** **Note** This is a required property only if the value of the property `pi.webgui.sso.oidc.client.authentication-method` is not set to NONE.
`pi.webgui.sso.oidc.client.authentication-method`	**Configure the OIDC client authentication method. The possible values are:** • **`BASIC` - Basic authentication header based client authentication** • **`POST` - Client credentials sent in POST body for authentication** • **`NONE` - Client does not authenticate itself** **The default value is `BASIC`.** **Note** If Client Authentication configuration in the OAuth client created in PingFederate is set to `NONE`, then use `NONE` for this property. If Client Authentication is set to `CLIENT SECRET`, use `BASIC` or `POST`. This is a mandatory property.
`pi.webgui.sso.oidc.provider.user-uniqueid-claim-name`	**The value of this property should be the claim name that holds the unique value to identify the signed-on user. It provisions a new user in the Dashboard data source or updates the user if it already exists with updated claim, if any. The default value in the Dashboard is `sub`.** **Note** This is a mandatory property.

Copyright © 2025 Ping Identity Corporation

Property	Description
`pi.webgui.sso.oidc.provider.user-first-name-claim-name`	The value of this property should be the claim name that holds the first name of the signed-on user. The default value for the claim is `given_name`. If you configure any other non-standard claim to send the value of first name in UserInfo, the name of that claim should be configured in the Dashboard properties as follows. **`pi.webgui.sso.oidc.provider.user-first-name-claim-name=my_first_name_claim`** **Note** This is a mandatory property.
`pi.webgui.sso.oidc.provider.user-last-name-claim-name`	The value of this property should be a claim name that holds the last name of the signed-on user. The default value for the claim is `family_name`. If you configure any other non-standard claim to send the value of last name in UserInfo, the name of that claim should be configured in Dashboard properties as follows: **`pi.webgui.sso.oidc.provider.user-last-name-claim-name=my_last_name_claim`** **Note** This is a mandatory property.
`pi.webgui.sso.oidc.provider.user-role-claim-name`	The value of this property should be a claim name that holds the value of the role of the signed-on user. The default value in the Dashboard is `role`. If the user uses a different claim name in PingFederate to send the role value, the same should be updated in this property. For example, `pi.webgui.sso.oidc.provider.user-role-claim-name=my_role_claim_name`. **Note** This is a mandatory property.
`pi.webgui.sso.oidc.client.additional-scopes`	The value of this property should be any additional scopes (comma separated) that need to be passed in the authorization request if required by the enterprise for retrieving the role claim. For example, `pi.webgui.sso.oidc.client.additional-scopes=read, read_role`. Such scopes, if any, should be created in PingFederate and attached to the OAuth client created in PingFederate for the Dashboard and configured to return the role claim for authorization in the Dashboard. This is not a mandatory property.

Copyright © 2025 Ping Identity Corporation

Example

The following is a sample snippet of `sso.properties`.

```
## PingIntelligence WebGUI SSO properties file
# This is in standard java properties file format
# comments are denoted by number sign (#) as the first non blank character
# multiline values are ended with '\' as end of line

# OIDC Provider uri
# WebGUI queries <issuer-uri>/.well-known/openid-configuration to get OIDC provider metadata
# issuer ssl certificate is not trusted by default. So import issuer ssl certificate into config/webgui.jks
# issuer should be reachable from both back-end and front-end
 pi.webgui.sso.oidc.provider.issuer-uri=https://localhost:9031

# OIDC Client id
pi.webgui.sso.oidc.client.id=PingIntelligence

# OIDC Client secret
# This can be empty
 pi.webgui.sso.oidc.client.secret=OBF:AES:BcB3MOE/K+VAa579oBpky4PrIo4z9LnI4vXsltqI=

# OIDC Client authentication mode.
# Valid values: BASIC, POST, and NONE
 pi.webgui.sso.oidc.client.authentication-method=BASIC

# claim name for unique id of the user in UserInfo response
# a new user is provisioned using this unique id value
 pi.webgui.sso.oidc.provider.user-uniqueid-claim-name=sub

# claim name for first name of the user in UserInfo response
# either first name or last name can be empty, but both should not be empty
 pi.webgui.sso.oidc.provider.user-first-name-claim-name=given_name

# claim name for last name of the user in UserInfo response
# either first name or last name can be empty, but both should not be empty
 pi.webgui.sso.oidc.provider.user-last-name-claim-name=family_name

# claim name for role of the user in UserInfo response
# valid values for roles are ADMIN,REGULAR
 pi.webgui.sso.oidc.provider.user-role-claim-name=role

# additional scopes in authorization request
# multiple scopes should be comma (,) separated
# openid,profile scopes are always requested
 pi.webgui.sso.oidc.client.additional-scopes=exclusive
```

Configuring SSO with PingOne

This topic discusses steps involved in configuring single sign-on (SSO) to the PingIntelligence for APIs Dashboard from PingOne. This feature is available in PingIntelligence for APIs 4.4.1 and later versions.

Before you begin

Copyright © 2025 Ping Identity Corporation

Verify the following prerequisites for SSO configuration:

- An installed PingIntelligence for APIs Dashboard.
- Access to the PingOne administration console console. For more information, see Accessing the admin console home page.

About this task

SSO configuration for the PingIntelligence Dashboard involves configuring both the Dashboard and PingOne.

Steps

1. Create an OIDC (OpenID Connect) web application in PingOne to set up SSO to the PingIntelligence Dashboard . To configure the OIDC application, complete the steps explained in Configuring an OIDC application in PingOne for PingIntelligence Dashboard.
2. Set the value of `pi.webgui.server.authentication-mode` to `sso` in `<pi_install_dir>/pingidentity/webgui/config/webgui.properties` file.

```
# Authentication mode
# valid values: native, sso
 pi.webgui.server.authentication-mode=sso
```

> **Note**
>
> The PingIntelligence for APIs Dashboard provides two methods for user authentication: native or SSO. SSO authentication should be used only for production deployments. Use native authentication for PoC deployments.

3. Configure the `<pi_install_dir>/pingidentity/webgui/sso.properties` file to complete the PingIntelligence Dashboard's SSO authentication. For more information, see Configuring Dashboard `sso.properties` for PingOne.
4. Obfuscate keys added in SSO properties using the following commands:

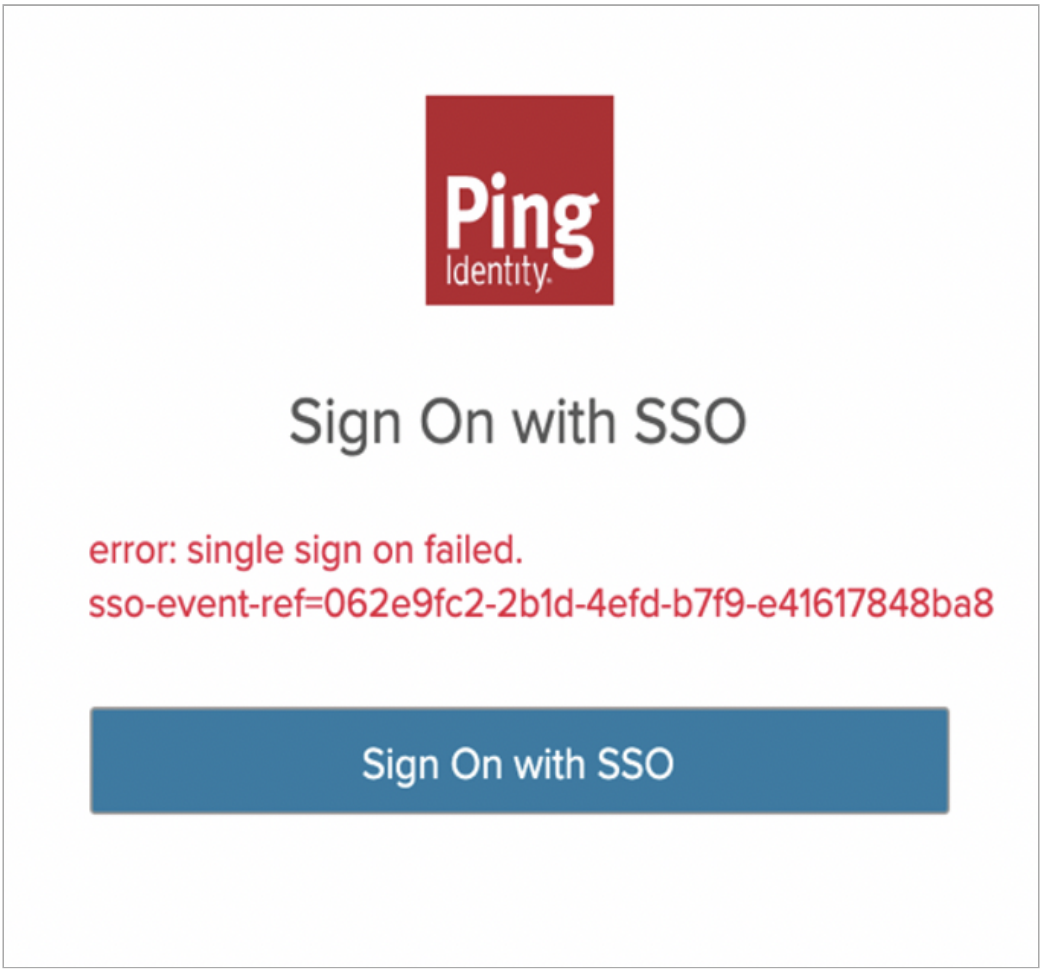
```
# cd <pi_install_dir>/pingidentity/webgui
# ./bin/cli.sh obfuscate_keys
```

5. Restart the PingIntelligence Dashboard after configuring SSO in PingOne and PingIntelligence Dashboard. For more information, see Starting and stopping thePingIntelligenceDashboard.
6. When the PingIntelligence Dashboard is started successfully, access it using `https://<pi_install_host>:8030`. The Dashboard will start SSO Authentication, and a new session will get created for the logged-in users.

Troubleshooting

If the SSO authentication fails for any reason, the PingIntelligence Dashboard shows the following error message.

Copyright © 2025 Ping Identity Corporation



Note

Every PingIntelligence Dashboard SSO authentication event is attached with a unique ID, which is logged in `<pi_install_dir>/pingidentity/webgui/logs/admin/sso.log`. You can filter `sso-event-ref = <unique ID>` in the `<pi_install_dir>/pingidentity/webgui/logs/admin/sso.log` file to find the reason for SSO failure.

Configuring an OIDC application in PingOne for PingIntelligence Dashboard

Complete the following steps in PingOne to create and configure an OpenID Connect (OIDC) application for setting up single sign-on (SSO) to PingIntelligence for APIs Dashboard.

Steps

1. **From the PingOne dashboard, create a new connection.**
2. **Provide the information for following fields:**
    - **Application Name**
    - **Description (Optional)**
    - **Icon (Optional)**

Copyright © 2025 Ping Identity Corporation

3. For Application Type, choose OIDC Web App, and click Save.
4. On the Configuration tab, click the Pencil icon.
5. Enter the following URL in the Redirect URLs field and click Save.

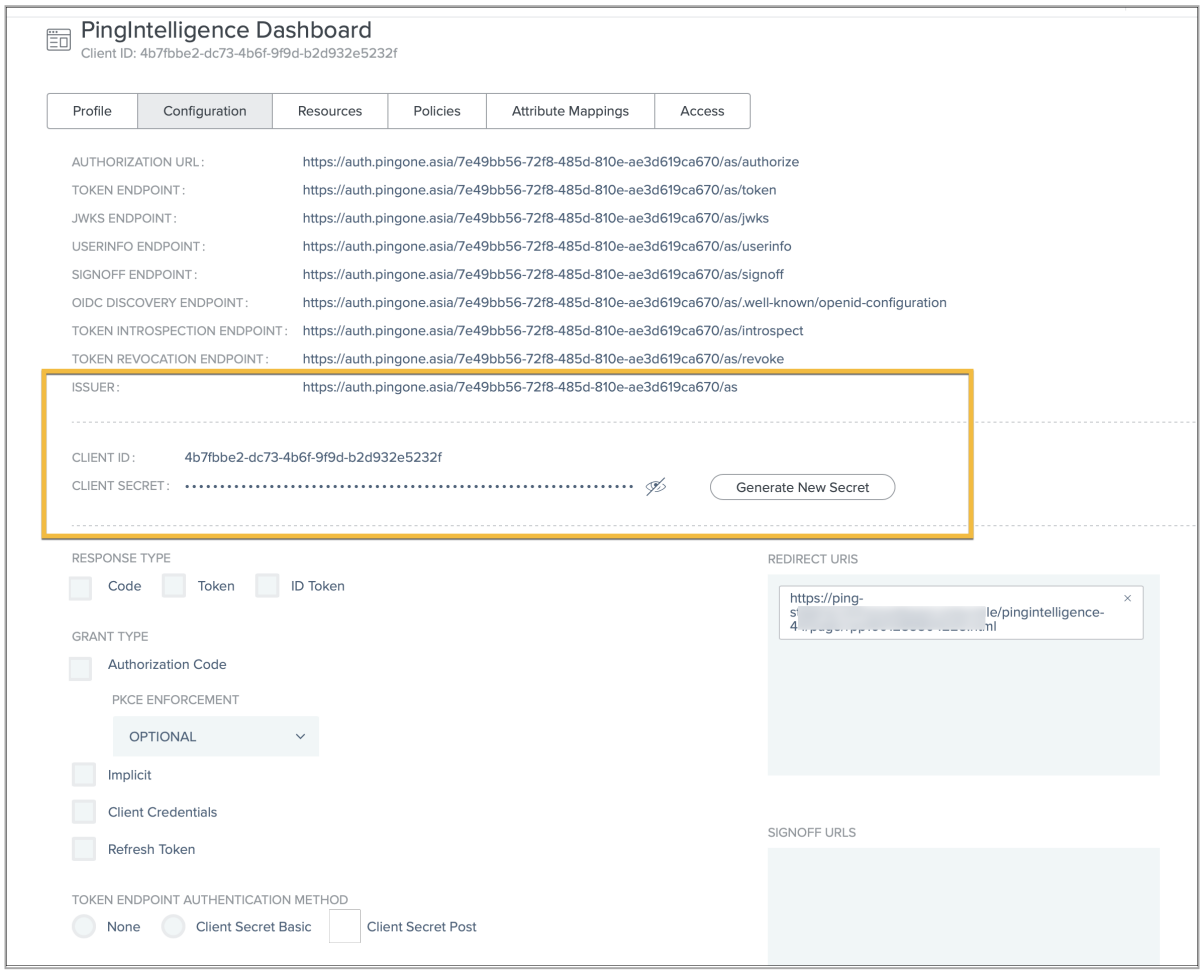
   The path in the URI is as follows: `https://<pi_install_host>:8030/login/oauth2/code/PingIntelligence`. Do not change the path in the URI, just substitute the hostname. For example, `https://127.161.140.180:8030/login/oauth2/code/PingIntelligence`.
6. Click the Resources tab and then click the Pencil icon. Click the + icon next to the profile scope to add to the list of Allowed Scopes. Click Save.
7. Click the Attribute Mappings tab and click the Pencil icon. Add the following attributes and map them to the PingIntelligence Dashboard `SSO.properties`. Click Add to add additional attributes. Make sure to select the Required check box for each attribute. When you are finished, click Save.

OIDC Attributes	Value
User ID PingOne User Attribute	The value defaults to `sub`.
Family Name PingOne User Attribute	The value of this property should be a claim name that holds the last name of the signed-on user in `<pi_installation_path>/pingidentity/webgui/sso.properties` file. The default value for the claim is `family_name`. For more information, see Configuring Dashboard `sso.properties` for PingOne.
Given Name PingOne User Attribute	The value of this property should be a claim name that holds the first name of the signed-on user in `<pi_installation_path>/pingidentity/webgui/sso.properties` file. The default value for the claim is `given_name`. For more information, see Configuring Dashboard `sso.properties` for PingOne.
Role Static Key	The value of this property should be a claim name that holds the value of the role of the signed-on user in `<pi_installation_path>/pingidentity/webgui/sso.properties` file. For more information, see Configuring Dashboard `sso.properties` for PingOne. [pingintelligence_configure_oidc_app_p1.dita] The default value in Dashboard is `role`. Supported values for the Role claim are `ADMIN` and `REGULAR`.

8. Click the Configuration tab and record the values for the following application properties to use in later steps in Configuring Dashboard `sso.properties` for PingOne:
   - Issuer
   - Client ID

Copyright © 2025 Ping Identity Corporation

- **Client Secret**



9. Click the Pencil icon on the top right, set the following properties, and click Save.

Property	Value
Response Type	Select Code.
Grant Type	Select Authorization Code. Keep the PKCE Enforcement as OPTIONAL.
Token Endpoint Authentication Method	Select None, Client Secret Basic, or Client Secret Post.

10. To enable the application, click the toggle switch to the on (blue) position.

Next steps

Complete the SSO configuration in PingIntelligence for APIs Dashboard. For more information see, Configuring Dashboard `sso.properties` for PingOne.

sso.properties for PingOne">

Copyright © 2025 Ping Identity Corporation

Configuring Dashboard `sso.properties` for PingOne

Configure the PingIntelligence Dashboard `sso.properties` for PingOne.

About this task

To complete the Dashboard's SSO authentication, configure the `<pi_installation_path>/pingidentity/webgui/sso.properties` file.

Steps

1. To complete the Dashboard's SSO authentication, configure the `<pi_installation_path>/pingidentity/webgui/sso.properties` file.

   The following table describes the SSO properties.

Property	Mandatory	Description
`pi.webgui.sso.oidc.provider.issuer-uri`	Yes	Configure the Issuer URI auto generate in PingOne for the PingIntelligence Dashboard application. For more information, see step 6 in Configuring an OIDC application in PingOne for PingIntelligence Dashboard.
`pi.webgui.sso.oidc.client.id`	Yes	Configure the client ID. Make sure to configure the same value auto generated in PingOne for the PingIntelligence Dashboard application. For more information, see step 6 in Configuring an OIDC application in PingOne for PingIntelligence Dashboard.
`pi.webgui.sso.oidc.client.secret`	This is a required property only if the value of the property `pi.webgui.sso.oidc.client.authentication-method` is not set to `NONE`.	Configure the client secret value in plain text. Make sure to configure the same value auto generated in PingOne for the PingIntelligence Dashboard application. For more information, see step-6 in Configuring an OIDC application in PingOne for PingIntelligence Dashboard.

Copyright © 2025 Ping Identity Corporation

Property	Mandatory	Description
`pi.webgui.sso.oidc.client.authentication-method`	Yes	Configure the PingOne OIDC application authentication method. The possible values are: ◦ `BASIC` - Basic authentication header-based client authentication. ◦ `POST` - Client credentials sent in POST body for authentication. ◦ `NONE` - Client does not authenticate itself. The default value is `BASIC`. **Note** If the Authentication method in the OIDC application created in PingOne is set to `NONE`, then use `NONE` for this property. If Authentication is set to Client Secret `BASIC`, Client Secret `POST`, use `BASIC` or `POST`.
`pi.webgui.sso.oidc.provider.user-uniqueid-claim-name`	Yes	The value of this property should be `sub`. It defaults to the value of User ID in PingOne OIDC Attributes.
`pi.webgui.sso.oidc.provider.user-first-name-claim-name`	Yes	The value of this property should be the PingOne OIDC Attribute value that holds the first name of the signed-on user. The default value for the claim is `given_name`.
`pi.webgui.sso.oidc.provider.user-last-name-claim-name`	Yes	The value of this property should be the PingOne OIDC Attribute value that holds the last name of the signed-on user. The default value for the claim is `family_name`.
`pi.webgui.sso.oidc.provider.user-role-claim-name`	Yes	The value of this property should be the PingOne OIDC Attribute value that holds the role of the signed-on user. The default value in Dashboard is `role`. Supported values for the Role claim are ADMIN and REGULAR.
`pi.webgui.sso.oidc.client.additional-scopes`	No	Not applicable for PingOne SSO configuration

The following is a sample snippet of `sso.properties`.

Copyright © 2025 Ping Identity Corporation

```
## PingIntelligence WebGUI SSO properties file
# This is in standard java properties file format
# comments are denoted by number sign (#) as the first non blank character
# multiline values are ended with '\' as end of line

# OIDC Provider uri
# WebGUI queries <issuer-uri>/.well-known/openid-configuration to get OIDC provider metadata
# issuer ssl certificate is not trusted by default. So import issuer ssl certificate into config/
webgui.jks
# issuer should be reachable from both back-end and front-end
 pi.webgui.sso.oidc.provider.issuer-uri=https://auth.pingone.asia/7e49bb56-72f8-485d-810e-
ae3d619ca670/as

# OIDC Client id
 pi.webgui.sso.oidc.client.id=PingIntelligence

# OIDC Client secret
# This can be empty
 pi.webgui.sso.oidc.client.secret=OBF:AES:BcB3MOE/K+VAa579oBpky4PrIo4z9LnI4vXsltqI=

# OIDC Client authentication mode.
# Valid values: BASIC, POST, and NONE
 pi.webgui.sso.oidc.client.authentication-method=BASIC

# claim name for unique id of the user in UserInfo response
# a new user is provisioned using this unique id value
 pi.webgui.sso.oidc.provider.user-uniqueid-claim-name=sub

# claim name for first name of the user in UserInfo response
# either first name or last name can be empty, but both should not be empty
 pi.webgui.sso.oidc.provider.user-first-name-claim-name=given_name

# claim name for last name of the user in UserInfo response
# either first name or last name can be empty, but both should not be empty
 pi.webgui.sso.oidc.provider.user-last-name-claim-name=family_name

# claim name for role of the user in UserInfo response
# valid values for roles are ADMIN and REGULAR
 pi.webgui.sso.oidc.provider.user-role-claim-name=role

# additional scopes in authorization request
# multiple scopes should be comma (,) separated
# openid,profile scopes are always requested
pi.webgui.sso.oidc.client.additional-scopes=exclusive
```

Next steps

Complete steps 4-6 Configuring SSO with PingOne.

Copyright © 2025 Ping Identity Corporation

Automatic rollover index

PingIntelligence for APIs Dashboard uses index lifecycle management (ILM) policy support of Elasticsearch to roll over time-series data.

Rolling over the time-series data is important to maintain a low latency during search operations. The ILM policy allows for an automatic rollover of index based on time or size of data.

> **Note**
>
> ILM policy for automatic rollover index works in Elasticsearch with X-Pack.

Configure automatic rollover index

You can configure the path to the ILM policy in `es.index.dashboard.activity.ilm.policy` property in `dashboard/config/dashboard.properties` file. The ILM policy file should be a valid JSON. Following is a sample `ilm.json` file available in the `dashboard/config` directory. Leave the value of `es.index.dashboard.activity.ilm.policy` property empty if you do not wish to use ILM policy.

Copyright © 2025 Ping Identity Corporation

```
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_size": "30GB",
            "max_age": "30d"
          },
          "set_priority": {
            "priority": 100
          }
        }
      },
      "warm": {
        "min_age": "30d",
        "actions": {
          "shrink": {
            "number_of_shards": 1
          },
          "readonly": {},
          "forcemerge": {
            "max_num_segments": 1
          },
          "set_priority": {
            "priority": 50
          }
        }
      },
      "cold": {
        "min_age": "90d",
        "actions": {
          "freeze": {},
          "set_priority": {
            "priority": 0
          }
        }
      }
    }
  }
}
```

Policy phases

The ILM policy is divided into three phases:

- `hot` - In the `hot` phase of the policy, the index is actively used to read and write data. The index remains in the `hot` phase till the defined policy age or if the index reaches the maximum size. After the index reaches the age or size, it is rolled over and new index is created.

  Configure the `max_age` and `max_size` of the rollover index. The index is rolled over based on which value among the size and age is triggered first.

- `warm` - In the `warm` phase of the policy, no new data is written to the index, however, it may be more frequently queried for searching data. The index next moves to the `cold` phase.

  Configure the `min_age` of the index for the `warm` phase.

Copyright © 2025 Ping Identity Corporation

- `cold` - In the `cold` phase, index is neither written to or read from. In the `cold` phase of policy, you can move the index to a low cost storage device.

  Configure the `min_age` of the index for the `cold` phase.

Priority

After an Elasticsearch restart, indices are reloaded back into memory in sequence according to priority. The index with the highest priority is loaded first. In the above sample JSON, the `hot` phase with priority 100 is of the highest priority. The `hot` index will be loaded into memory first. The `warm` phase with a priority number 50 is second in priority. The `warm` index will be loaded into memory after the `hot` index. Use a positive integer number to set the priority.

Related links

- https://www.elastic.co/guide/en/elasticsearch/reference/6.8/index-lifecycle-management.html
- https://www.elastic.co/guide/en/elasticsearch/reference/6.8/ilm-policy-definition.html

Splunk for PingIntelligence

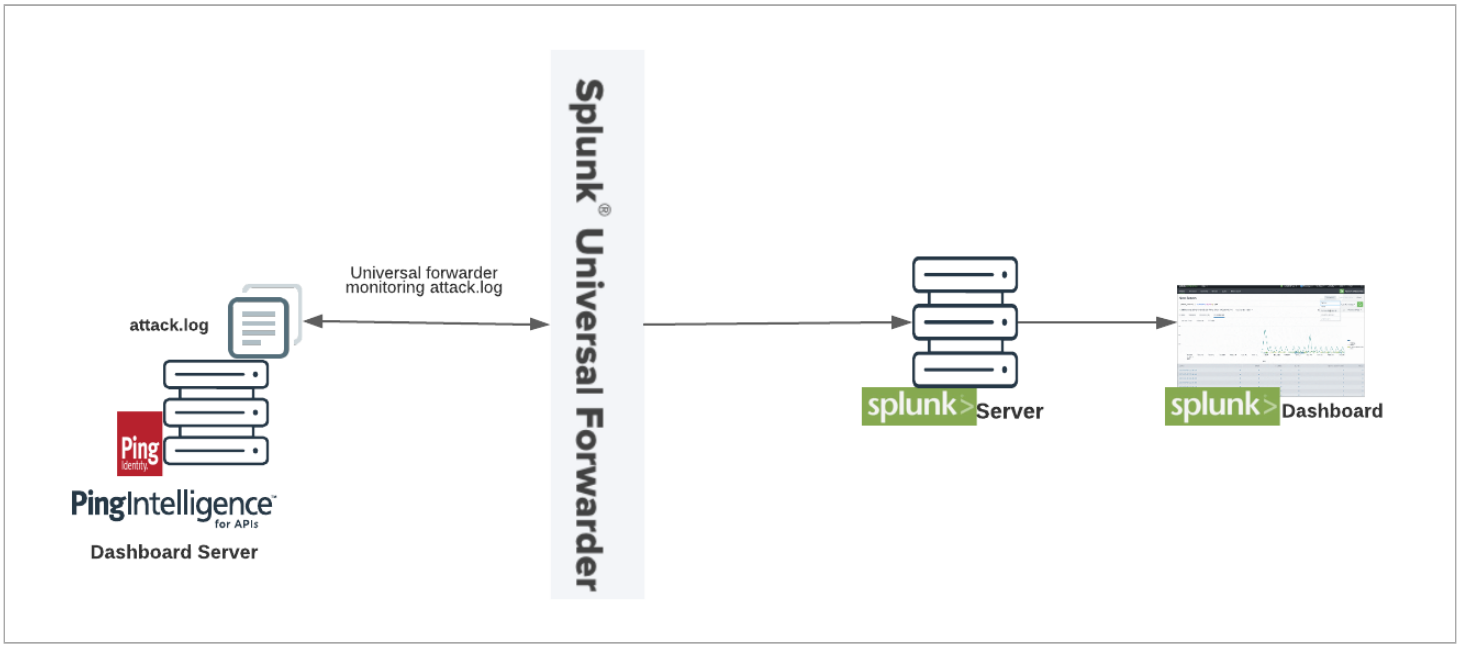
Splunk for PingIntelligence provides a pictorial view of various attacks in an API environment with granular event details.

The Splunk Dashboard monitors the `attack.log` file in PingIntelligence for APIs Dashboard. The Dashboard server through `attack.log` returns a JSON report that contains attack details. The following is a snippet of `attack.log` with attack details:

```
{
  "timestamp": "1575965866132",
  "protocol": "HTTP",
  "attack_id": "11",
  "description": "Extreme App Activity",
  "attack_bucket": "API",
  "attack_scope": "SINGLE_API",
  "attacked_api": "shop-electronics",
  "attack_identifier_type": "TOKEN",
  "attack_key": "",
  "attack_value": "343077883101e1c8f2b3ec0fbf6a32ab2327e4c2e7ebe525a27a125225fa136d"
}
```

The following illustration summarizes the data flow between the PingIntelligence Dashboard and Splunk.

Copyright © 2025 Ping Identity Corporation



Note

PingIntelligence for APIs is qualified for Splunk 8.0.0.

Installing and configuring Splunk for PingIntelligence

To complete the configuration of Splunk for PingIntelligence, you need to create a source type.

About this task

Creating a source type helps Splunk to understand the event format. The source type is one of the default fields that Splunk assigns to all the incoming data. Configuring the source type informs Splunk about the type of data ABS provides. This helps Splunk in formatting data intelligently during indexing.

To create a source type, complete the following steps:

Steps

1. Configure a new source type by navigating to Splunk Enterprise → Settings → Source Types → New Source Type.

   Result:

   The Source Type Events page is displayed.

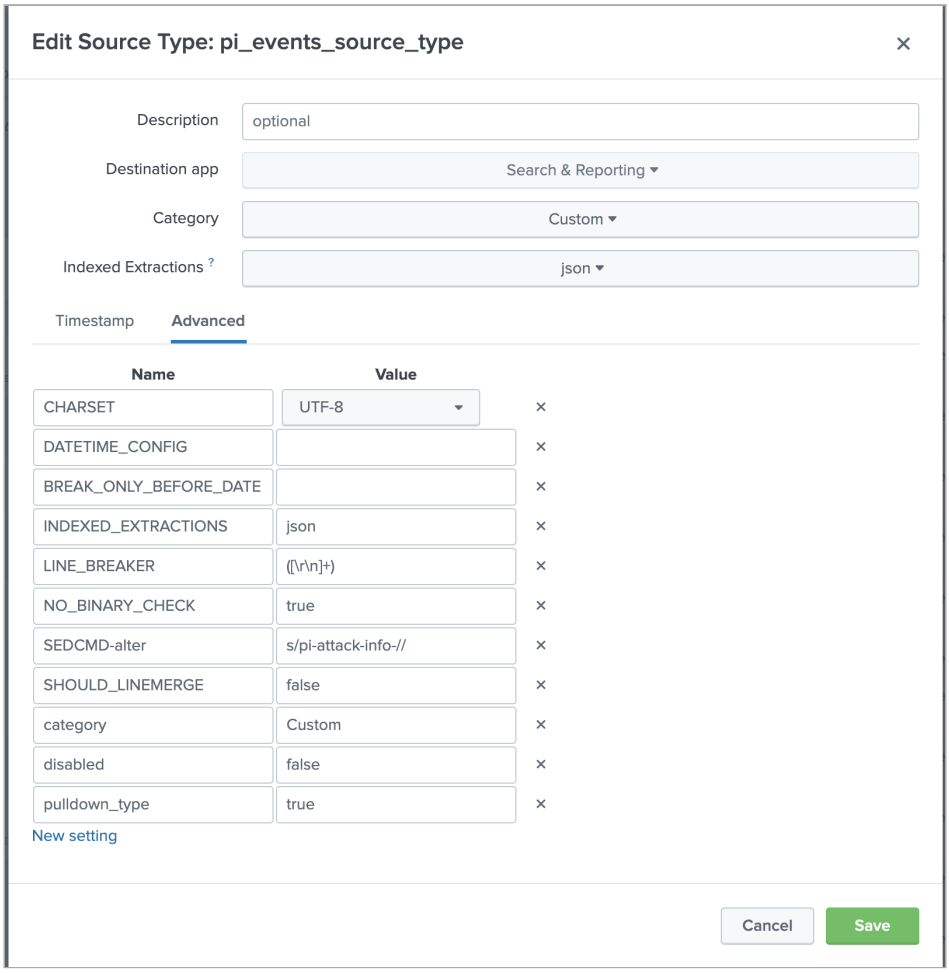
2. Configure the New Source Type.

   The fields are defined in the following table.

Name	Value
Source Type Name	`pi_events_source_type`

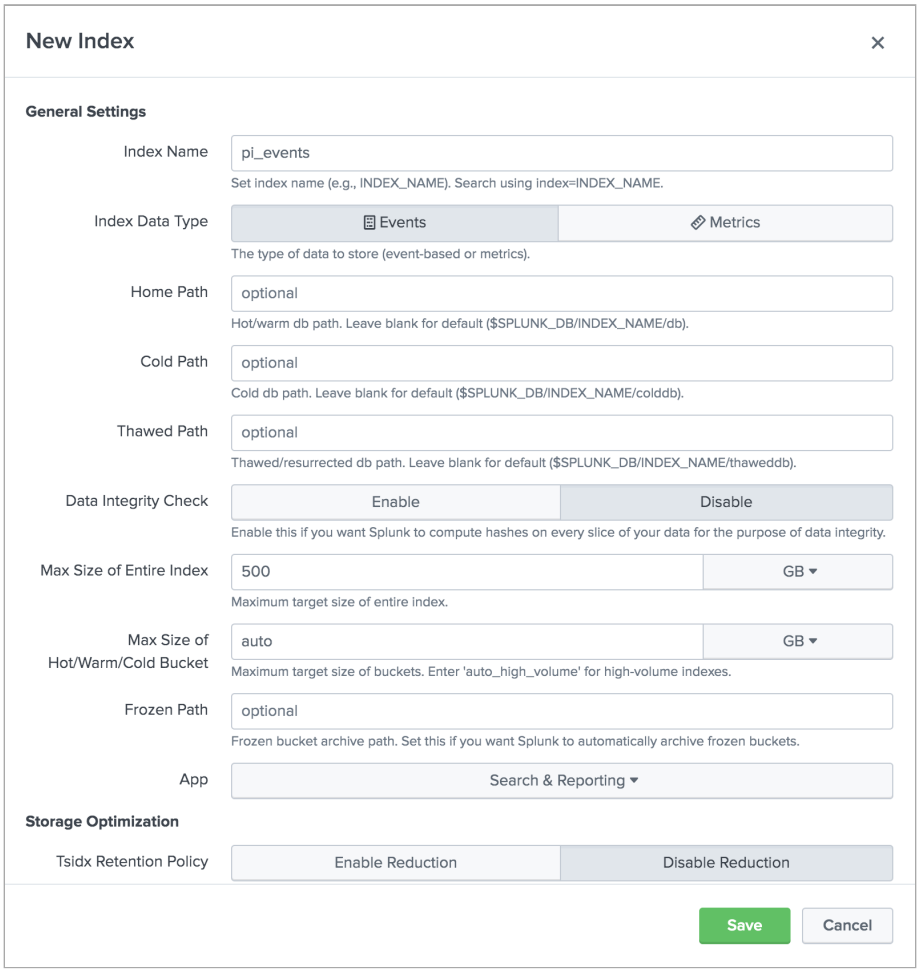
Copyright © 2025 Ping Identity Corporation

Name	Value
Destination app	Search and Reporting (Can change for your apps)
Category	Structures
Indexed Extractions	`json`
SEDCMD-alter	`s/pi-attack-info-//`



3. Create a new index `pi_events` by navigating to Enterprise → Settings → Indexes → New Index.

Copyright © 2025 Ping Identity Corporation



Types of data captured

Splunk for PingIntelligence captures attack data.

The attack event captures the components listed in the following table:

Field	Description
timestamp	epoch timestamp
protocol	HTTP(s) /Websocket (ws)
attack_id	PingIntelligence attack ID
description	Description of the attack.
attack_bucket	Attack on an API or a DDoS attack.
attack_scope	Single or multiple APIs.
attacked_api	Name of the API. In case of multiple APIs, MULTI_API is reported.

Copyright © 2025 Ping Identity Corporation

Field	Description
attack_identifier_type	Username, API Key, OAuth token, Cookie, or IP address.
attack_key	Details of APIKEY or Cookie.
attack_value	Value of the client identifier.

Installing and configuring the Splunk Universal Forwarder

Install and configure the Splunk Universal Forwarder to collect attack data and forward it to the Splunk server.

About this task

To install and configure Splunk Universal Forwarder:

Steps

1. Download Splunk Universal Forwarder 8.0.0. For more information, see Splunk® Universal Forwarder Manual.
2. Install the Splunk Universal Forwarder by entering the following command:

```
[root@ABS]# tar -xvf splunkforwarder-8.0.0-8c86330ac18-Linux-x86_64.tgz
splunkforwarder/
splunkforwarder/share/
```

> **Note**
>
> Replace the file name given in the example command with the name of the file you downloaded in step 1.

3. Start the Splunk Universal Forwarder.

```
[root@ABS]# cd splunkforwarder/bin
[root@ABS]# ./splunk start --accept-license
```

4. Add forward server details (the receiver host and port in Splunk).

   Example:

```
[root@dashboard]# ./splunk add forward-server ip:port

Splunk username: admin Password: Added forwarding to: 192.168.1.158:9997.
```

> **Note**
>
> Enable the receiving port in Splunk. For example, configure port number 9997 from the previous example in your Splunk deployment.

5. Edit the `inputs.conf` file on your Splunk Universal Forwarder as shown in the following example.

Copyright © 2025 Ping Identity Corporation

Example:

```
[root@ABS]# ./splunk add monitor /opt/pingidentity/splunk/data/
Added monitor of '/opt/pingidentity/splunk/data/'.
```

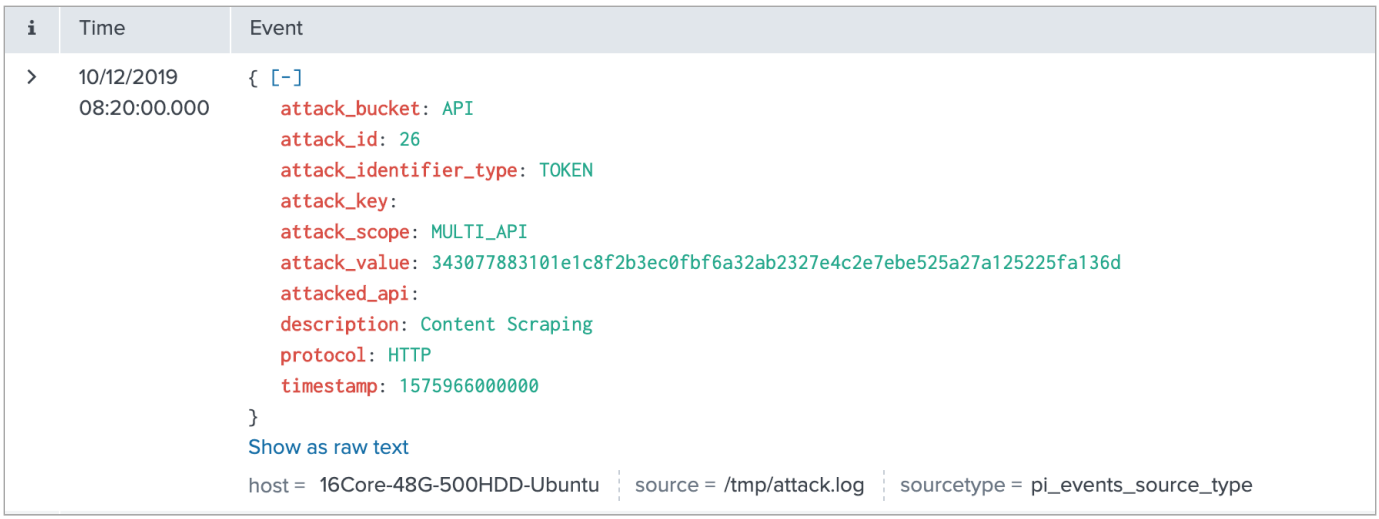
6. Edit the `inputs.conf` file on your Splunk Universal Forwarder.

```
[root@dashboard]# cat /opt/splunkforwarder/etc/apps/search/local/inputs.conf

[monitor:///opt/pingidentity/pingidentity/dataengine/logs/attack.log/]

index = pi_events
sourcetype=pi_events_source_type
disabled = false
```

7. Restart the Splunk Universal Forwarder.

```
[root@ABS]# ./splunk restart
```

8. Verify if data is flowing to Splunk on the Splunk Dashboard.



Troubleshooting:

If no data is available in Splunk, check your firewall settings.

Configuring alert notifications on Slack and email

You can configure Splunk to send alert notifications to a Slack channel or through email.

Before you begin

Make sure to install the Slack app in your Splunk setup. Also be sure to connect Slack and Splunk using webhooks. For more information on Slack webhooks, see Incoming Webhooks.

About this task

Copyright © 2025 Ping Identity Corporation

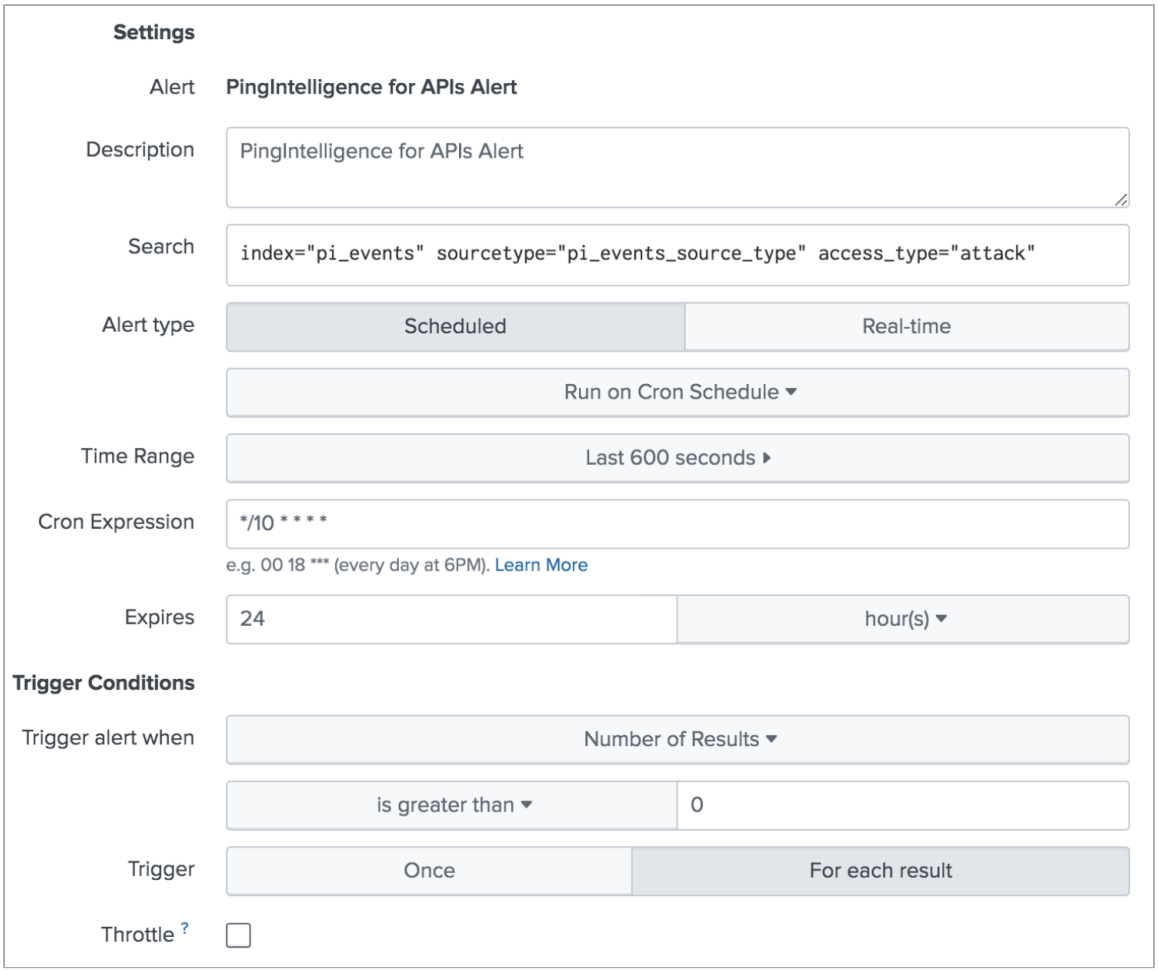
Complete the following steps to create an alert for Slack:

Steps

1. **Navigate to Settings -> Searches, reports and alerts.**

> **Note**
>
> Alerts should be created for App: Search & Reporting(search).

2. **Create new alerts. Enter the values as described in the table below:**

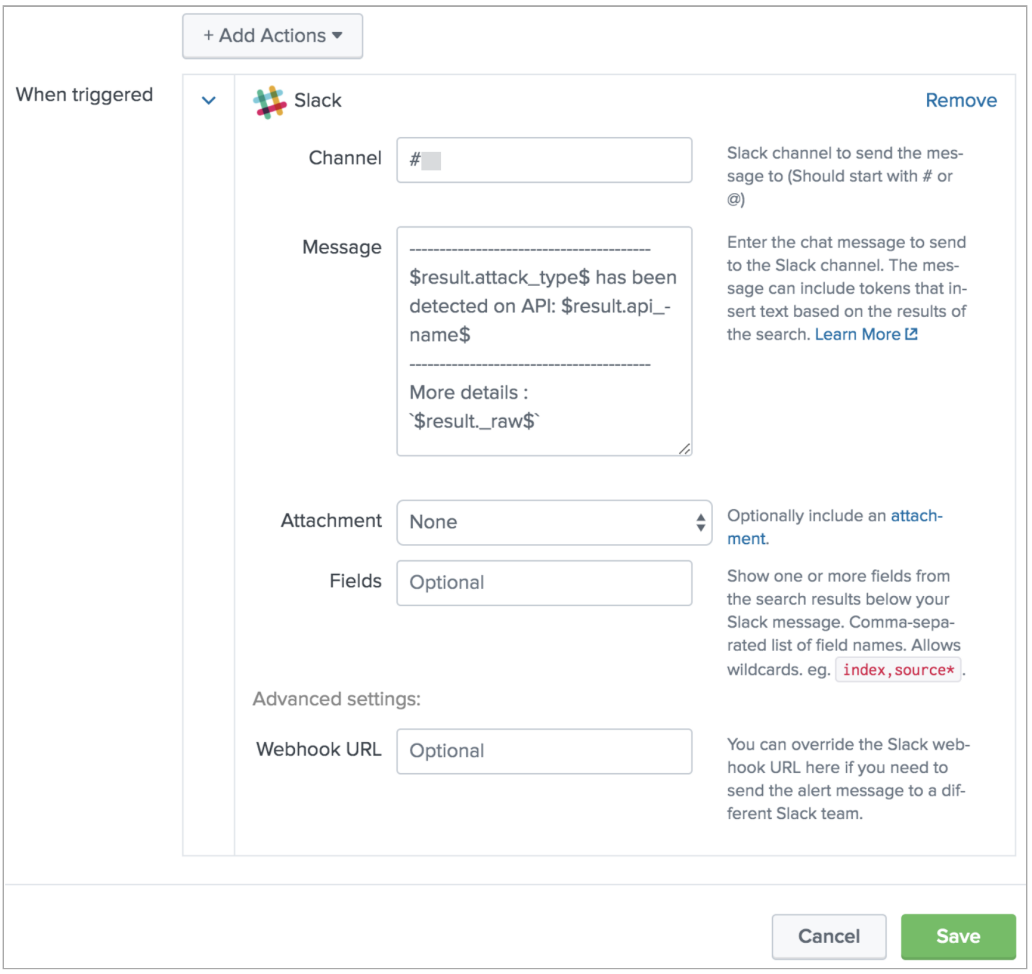


Value	Description
Description	`PingIntelligence for APIs Alert`
Search	`Search: index="pi_events" sourcetype="pi_events_source_type" access_type="attack"`
Alert Type	**Scheduled → Run on Cron Schedule**
Time Range	**Last 600 seconds**

Copyright © 2025 Ping Identity Corporation

Value	Description
Cron Expression	`*/10 * * * *`
Expires	`24 Hours`
Trigger alert when	**The alert should be triggered for results when greater than 0.**
Trigger	**For each result. This would trigger a new alert for each event.**
Throttle	**Do not throttle the events.**

3. **Configure alert action as follows.**



Value	Description
Add Actions	**Choose the Slack app to add actions.**

Copyright © 2025 Ping Identity Corporation

Value	Description
Channel	Use the channel that has been configured with a Webhook URL starting with either # or @. In this example, we are using Channel name as: `# PingIntelligence_alerts`

Message	This is the message that will be posted along with the alert in Slack. We recommend using the following message:
	`----------`
	`$result.attack_type$ has been detected on API: $result.api_name$`
	`----------`
	`More details :`
	`$result._raw$`

Attachments	N/A
Fields	N/A
Webhook URL	N/A

4. Post a message in Splunk to verify that it is notified in Slack.

attack.log for Splunk">

Configuring `attack.log` for Splunk

Configure `attack.log` for Splunk to capture attack data.

About this task

To configure `attack.log`:

Steps

1. Configure `dataengine.properties` for `attack.log`.
    1. Edit the `pingidentity/dataengine/config/dataengine.properties` file to send the attack data to `attack.log`. By default, `syslog` is configured.
    2. To send the attack data to `attack.log`, edit the `dataengine.properties` file as shown in the snippet below:

Copyright © 2025 Ping Identity Corporation

```
# Log4j2
# publish attacks to Log4j2. Valid values true or false
# By default it provides syslog support
 publish.log4j2.enable=true
# log4j2 config file to log attacks to an external service. For example, Syslog
# use com.pingidentity.abs.publish as logger name in log4j2 configuration
 log4j2.config=config/attack_log.xml
# log4j2 log level for attack logging
log4j2.log.level=INFO
# directory for any log4j2 config dependency jar's.
# useful for third party log4j2 appenders
# it should be a directory
log4j2.dependencies.dir=plugins/
```

Result:

The following is a snippet of the `attack_log.xml`. The `attack_log.xml` produces `attack.log` that is consumed by Splunk. The `attack.log` captures the attack data in a JSON format.

The attack data is published to a Log4j logger named `com.pingidentity.abs.publish`. The Log4j configuration file must have a logger named `com.pingidentity.abs.publish`. Any Log4j2 config file that wants to capture attack data from Dashboard must have at least one logger with the name `com.pingidentity.abs.publish`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration name="APIIntelligence" packages="" status="warn">
  <Appenders>
    <RollingFile name="attack_log" append="true" fileName="${sys:dashboard.rootdir}/logs/attack.log"
      filePattern="logs/attack.log.%d{yyyy-MM-dd}" immediateFlush="true" >
      <PatternLayout>
        <Pattern>pi-attack-info-%m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy/>
      </Policies>
    </RollingFile>
  </Appenders>

  <!-- Attacks are logged to logger with name com.pingidentity.abs.publish
       There should be at least one logger with name com.pingidentity.abs.publish
       It is better to set additivity="false" so that same attacks will not be logged in dashboard.log
-->

  <Loggers>
    <Logger additivity="false" level="info" name="com.pingidentity.abs.publish">
      <AppenderRef ref="attack_log"/>
    </Logger>
  </Loggers>
</Configuration>
```

Dashboard log messages

The following table lists the critical log messages from the `dashboard.log` file.

Copyright © 2025 Ping Identity Corporation

The `dashboard.log` file is rotated every 24 hours.

Log messages	Description
`error - fatal protocol violation`	This message is logged in `dashboard.log` when there is an HTTP/(S) protocol error while connecting to API Behavioral Security (ABS) or Elasticsearch.
`error - fatal transport error`	This message is logged in `dashboard.log` when there is an unknown host for ABS or Elasticsearch.
`error - error while sending message to syslog`	This message is logged in `dashboard.log` when the `syslog` server is not reachable, or there is an error in configuration of Secure Sockets Layer (SSL) or non-SSL connections.
`error - capacity full in syslog consumer worker, retries exhausted, ignoring this message`	This message is logged in `dashboard.log` when the `syslog` server is not reachable, or there is an error in the configuration of SSL or non-SSL connections.
`error - error while closing response stream`	This message is logged in `dashboard.log` when ABS, or Elasticsearch socket is not closed properly.
`error - error while flushing file stream`	This message is logged in `dashboard.log` when there is a failure in the storage disk, or the storage disk is full.
`error - error while closing file stream`	This message is logged in `dashboard.log` when there is a failure in the storage disk, or the storage disk is full.
`error - error while parsing access_time from file`	This message is logged in `dashboard.log` when ABS returns an invalid `access_time`, or the time format is not consistent.
`error - error while parsing api_key name from file`	This message is logged in `dashboard.log` when ABS returns an empty API key in the API key metrics or attack report.
`error - error while parsing cookie name from file`	This message is logged in `dashboard.log` when ABS returns an empty cookie name in the metrics or attack report.
`warn - http request " + <URL> + ", response status: " + <Response status>`	This message is logged in `dashboard.log` when ABS or Elasticsearch returns an HTTP status code that is greater than or equal to 300.
`Dashboard stopped`	This message is logged in `dashboard.log` when the Dashboard is shut down.

Copyright © 2025 Ping Identity Corporation

Purging Dashboard logs

The `purge.sh` script either archives or purges processed access log files that are stored in the `/opt/pingidentity/dataengine/logs` directory.

About this task

Located in the `/opt/pingidentity/dataengine/util` directory, the `purge` script deletes logs older than the specified number of days.

> **Note**
>
> The number of days specified should be between 1-365 days.

Steps

1. Run the `purge` script using the Dashboard command line.

   Example:

   ```
   /opt/pingidentity/dataengine/util/purge.sh -d 3
   In the above example, purge.sh deletes all access log files older than 3 days. Here is sample
   output.
   /opt/pingidentity/dataengine/util/purge.sh -d 3
   This will delete the data in /opt/pingidentity/dataengine/logs which is older than 3 days.
   Are you sure (yes/no): yes
   removing /opt/pingidentity/dataengine/logs/dataengine.log.2019-02-07 : last changed at Sat Feb  9
   00:29:43 EST 2019
   removing /opt/pingidentity/dataengine/logs/dataengine.log.2019-02-09 : last changed at Mon Feb 11
   00:29:48 EST 2019
   removing /opt/pingidentity/dataengine/logs/dataengine.log.2019-02-08 : last changed at Sun Feb 10
   00:29:56 EST 2019
   Done.
   ```

   > **Note**
   >
   > When the `purge` script is run, the log files are permanently deleted from the `/opt/pingidentity/dataengine/logs` directory. Always back up the files before deleting.

2. To force delete the Dashboard log files:

   1. Use the `-f` option with the `purge.sh` script.

   When using this option, the script does not check for confirmation to purge the log files.

   1. Use the force purge option with the `-d` option to provide the number of days of logs to keep.

      Example:

      The following snippet shows an example of the force purge and `-d` option:

Copyright © 2025 Ping Identity Corporation

```
/opt/pingidentity/dataengine/util/purge.sh -d 3 -f
removing /opt/pingidentity/dataengine/logs/dataengine.log.2019-02-07 : last changed at Sat
Feb  9 00:31:26 EST 2019
removing /opt/pingidentity/dataengine/logs/dataengine.log.2019-02-09 : last changed at Mon Feb
11 00:31:30 EST 2019
removing /opt/pingidentity/dataengine/logs/dataengine.log.2019-02-08 : last changed at Sun Feb
10 00:31:35 EST 2019
Done.
```

3. To archive logs older than the specified number of days to secondary storage, use the `-l` option with the `purge.sh` script and include the path of the secondary storage to archive log files

   Example:

   In this example, log files older than `3-days` are archived to the `tmp` directory. To automate log archival, add the script to a `cron` job.

   ```
   /opt/pingidentity/dataengine/util/purge.sh -d 3 -l /tmp/
   ```

Purging data from Elasticsearch

To manage storage on the Dashboard server, you can either archive or purge Elasticsearch data.

About this task

PingIntelligence provides a purge script to remove older Elasticsearch data.

> **Warning**
>
> When the purge script is run, all files are permanently deleted from the Elasticsearch data directory. Hence it is recommended to make a backup of Elasticsearch documents before proceeding with the purge.

Steps

1. Run the purge script on the Dashboard engine command line.

   The number of days specified should be between 1-365 days.

   ```
   /opt/pingidentity/dashboard/util/purge_elasticsearch.sh -d 3
   ```

   Example:

   In the following example, `purge_elasticsearch.sh` deletes all files older than 3 days. Below is a sample output:

   ```
   /opt/pingidentity/dashboard/util/purge_elasticsearch.sh -d 3
   This will delete the data in elastic search which is older than 3 days.
   Are You sure(yes/no):yes
   2017-04-17 11:13:07 INFO Starting purge with options, days : 3 path : /opt/poc/pingidentity/
   dashboard/config/dashboard.properties
   ```


Copyright © 2025 Ping Identity Corporation

2. To delete all data and Elasticsearch templates, use the following:

```
curl -s https://<elasticsearch_ip_address>:<port>/_all -X DELETE -u elastic
```

Example:

The following example illustrates deletion of Elasticsearch records older than 15 days. The `Number of Records Purged : null` is an expected message due to the time lag in actual deletion.

```
[xxxxxxxxx@T5-03 dashboard]$ ./util/purge_elasticsearch.sh -d 15
This will delete the data in elasticsearch cluster which are older than 15 days.
Are You sure(yes/no):yes
Starting Elasticsearch purge
2020-04-09 03:16:44 INFO   Starting purge with options, days : 15 path : /home/xxxxxxxx/
pingidentity/dashboard/config/dashboard.properties
2020-04-09 03:16:45 INFO   API's Loaded from elasticsearch : [app54, app58, app63, app8, app2, app3,
app66, app74, app79, app77]
2020-04-09 03:16:45 INFO   Purging data for global indice activity-api
2020-04-09 03:16:45 INFO   Number of Records Purged :  null
2020-04-09 03:16:45 INFO   Purging data for global indice activity-api-key
2020-04-09 03:16:45 INFO   Number of Records Purged :  null
2020-04-09 03:16:45 INFO   Purging data for global indice activity-token
2020-04-09 03:16:45 INFO   Number of Records Purged :  null
.
.
```

Result:

When you use the `-X DELETE` option, the system goes back to a fresh installation state.

> **Note**
>
> Purge for Elasticsearch runs in the background. Documents are not deleted immediately after **`purge_elasticsearch.sh`** execution. Elasticsearch deletes purged documents with a lag of 5 minutes. It is recommended to run **`purge_elasticsearch.sh`** during lean API traffic periods.

Purging WebGUI logs

The `purge.sh` script either archives or purges processed access log files and admin log files that are stored in the `/opt/pingidentity/webgui/logs/access/` and `/opt/pingidentity/webgui/logs/admin/` directories respectively.

About this task

Located in the `/opt/pingidentity/webgui/util` directory, the `purge` script deletes logs older than the specified number of days.

> **Note**
>
> The number of days specified should be between 1-365 days.

Copyright © 2025 Ping Identity Corporation

Steps

1. Run the script using the WebGUI command line.

   Example:

   ```
   /opt/pingidentity/webgui/util/purge.sh -d 1
   This will delete the logs in /opt/e2e/pingidentity/webgui/logs/admin and /opt/e2e/pingidentity/
   webgui/logs/access that are older than 1 days.
   Are you sure (yes/no): yes
   Removing /opt/e2e/pingidentity/webgui/logs/admin/admin.log.2020-04-08 : last changed at Wed Apr  8
   17:07:49 UTC 2020
   removing /opt/e2e/pingidentity/webgui/logs/access/access.log.2020-04-08 : last changed at Wed Apr  8
   19:03:31 UTC 2020
   Done
   ```

   > **Note**
   >
   > When the **purge** script is run, the log files are permanently deleted. Hence it is recommended to always back up the files before deleting.

2. To force delete the WebGUI files:

   1. Use the `-f` option with the `purge.sh` script.

   When using this option, the script does not check for confirmation to purge the log files.

   1. Use the force purge option with the `-d` option to provide the number of days of logs to keep.

      Example:

      The following snippet shows an example of the force purge and `-d` option. In this example, the script force purges the WebGUI log files while keeping log files of 2 days.

      ```
      /opt/pingidentity/webgui/util/purge.sh -d 2 -f
      ```

3. To archive logs older than the specified number of days to secondary storage, use the `-l` option the `purge.sh` script and include the path of the secondary storage to archive log files.

   Example:

   In this example, log files older than `2-days` are archived to the `backup` directory. To automate log archival, add the script to a `cron` job.

   ```
   /opt/pingidentity/webgui/util/purge.sh -d 2 -l /backup/
   ```

Use Case: Converting SSL certificates to ASE compatible format

This topic discusses the commands involved in converting your SSL certificates to make them compatible with API Security Enforcer (ASE)'s SSL certificate format.

Copyright © 2025 Ping Identity Corporation

About this task

When PingIntelligence for APIs is deployed in sideband mode, ensure that the SSL certificates used by the gateway is in `.pem` format. You can use OpenSSL to convert the certificates.

To convert your SSL certificate from `.crt` extension to `.pem` extension:

Steps

1. Run the following command to get ASE certificate details:

   ```
   # openssl s_client -showcerts -connect  <ASE-IP>:<SSL-PORT>
   ```

   Example:

   ```
   openssl s_client -showcerts -connect 127.1.1.1:8443
   ```

2. Create a temporary certificate file `ase.crt` using the contents of the ASE certificate.

   > **Important**
   >
   > Make sure to include the content starting from "-----BEGIN CERTIFICATE-----" to "-----END CERTIFICATE-----" in the temporary `ase.crt` file.

3. Run the following command to convert the `ase.crt` certificate into a `.pem` file:

   ```
   # openssl x509 -in ase.crt -out ase.pem
   ```

Copyright © 2025 Ping Identity Corporation