PingIntelligence

Contents

PingIntelligence for APIs Overview

Digital transformation initiatives founded on APIs are making business logic and data readily accessible to internal and external users. However, APIs also present a new opportunity for hackers to reach into data and systems, and predefined rules, policies and attack signatures can't keep up with this evolving threat landscape. *PingIntelligence for APIs* uses artificial intelligence (AI) to expose active APIs, identify and automatically block cyberattacks on APIs, and provide detailed reporting on all API activity. Leveraging AI models specifically tailored for API security, PingIntelligence for APIs brings cyberattack protection and deep API traffic insight to existing API Gateways and application server-based API environments.

PingIntelligence for APIs detects anomalous behavior on APIs, as well as the data and applications exposed via APIs, and can automatically block attacks across your API environment. For example, attempts to bypass login systems using botnet credential stuffing attacks or stolen tokens are recognized as cyberattacks. Attempts to exfiltrate, change or delete data that fall outside the range of normal behavior for an API can also be blocked and reported on in near real time.

PingIntelligence for APIs Release Notes

PingIntelligence 4.2 Release Notes

New in PingIntelligence Dashboard

PingIntelligence 4.2 Dashboard has the following updates:

- **Attack Insight** - Attack insight provides information on why the AI engine marked a client identifier as a potential attacker. For more information, see *Attack insights* on page 465.
- **Client Forensic** - Client forensics provides insight into client activity in the course of an attack by presenting the client activity leading up to the attack. For more information, see *Client forensic report* on page 461.

New in ABS AI Engine

Enhanced attack detection - ABS AI engine was tuned to deliver improved detection of attacks on REST APIs. For more information, see *REST API attack types* on page 331.

New in API Gateway Integration Policies

- **Apigee API gateway policy update** - The updated PingIntelligence sideband policy supports improved performance when using the timeout options between Apigee API gateway and PingIntelligence ASE. For more information, see *Configure Apigee properties file* on page 515.
- **Azure API gateway policy update** - The updated PingIntelligence sideband policy supports capturing OAuth2 tokens from the query string. For more information, see *Deploy PingIntelligence policy* on page 577 for Azure API gateway.
- **Mulesoft API gateway policy update** - The updated PingIntelligence sideband policy includes an improved user interface for applying sideband policies in Mulesoft V4.x environments. For more information on applying PingIntelligence policy, see *Apply PingIntelligence policy* on page 627.
- **PingAccess API gateway policy update** - The updated PingIntelligence sideband policy adds optional asynchronous communication between PingAccess and ASE for improved performance when deployed in environments that do not require automated client blocking. It also supports capturing OAuth tokens from the query string. The policy now supports both PingAccess 5.x and 6.x. For more information, see *PingAccess sideband integration* on page 665.

Resolved Tickets

Following tickets have been resolved in PingIntelligence 4.2 release:

Ticket ID	Description
ABS AI engine - LML-233	An issue related to authorization header with bearer token is now resolved.
ABS AI engine - LML-222	An issue related to `/tmp` directory permission for ML jobs is now resolved.
Mulesoft 4.x API gateway - PIPOLICY-23	An issue related to empty USERNAME KEY and CLIENT ID KEY fields in the "Apply policy" page is now resolved.
ASE - BAL-276	An issue related to malformed authorization header is now resolved.
Dashboard - PIWEBAPI-11	An issue related to Cross Site Scripting attacks is now resolved. (Content Security Policy)

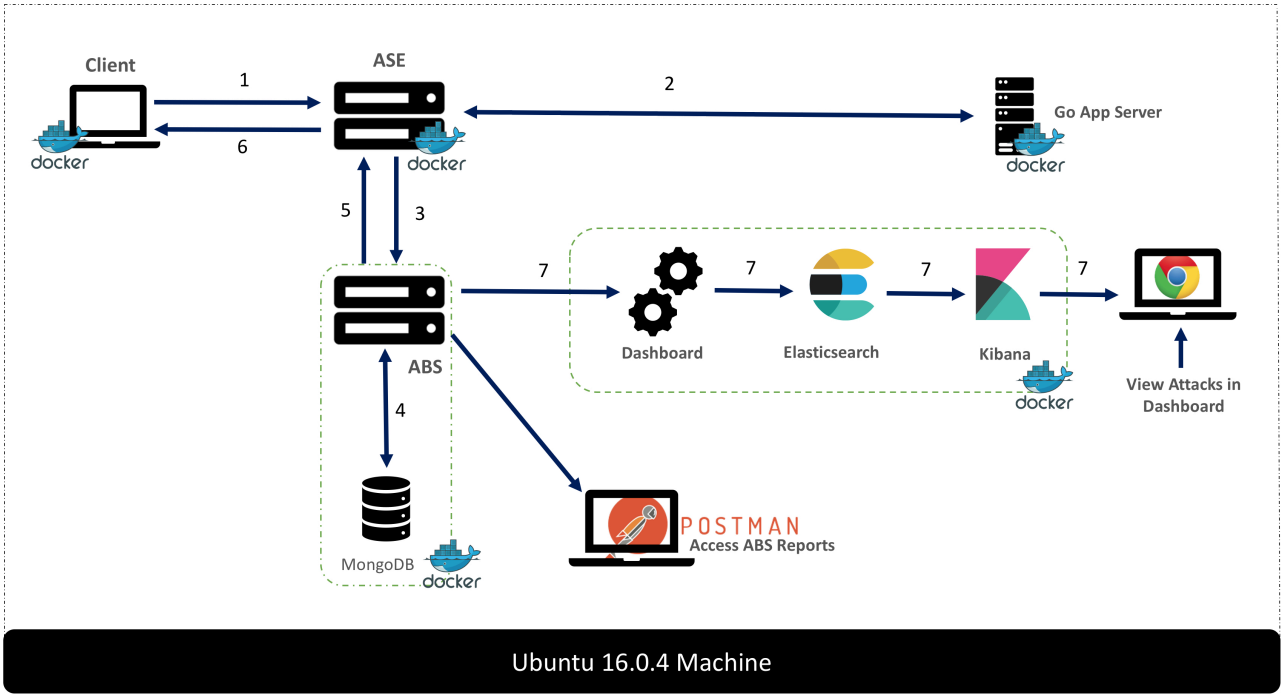
PingIntelligence PoC

PingIntelligence Docker PoC deployment

Docker PoC setup

This guide describes the installation and execution of PingIntelligence for APIs software in a Docker environment for inline and sideband deployment. The automation script imports and installs the Docker images. A script is run to generate normal API traffic to train the AI engine. After training is complete, another script is run to send a mixture of normal and attack traffic. The guide then explains how to access a graphical dashboard which shows activity on the test environment and detailed reporting on the API activity.

This Docker Evaluation Guide provides instructions for deploying a test configuration as shown in the diagram. The docker setup can be deployed in an inline mode where the client traffic directly reaches 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* on page 144 and *Inline ASE* on page 185.



ⓘ **Note:** The Docker images provided are only for evaluation purpose of PingIntelligence for APIs product. They should not be used in production deployments or for setting up environments for security testing.

Installation requirements

Here is a summary of the software, documentation, and server requirements:

Docker images

Download the Docker PoC 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
5. Google Go App server

ASE and ABS license: ASE and ABS licenses are required to start both the products. Contact the PingIntelligence team to access the trial license.

Postman reporting

ABS generates various REST API reports. You can view these reports using 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

Refer the following Admin Guides:

- ***ASE Admin Guide*** - Refer the ASE admin guide to learn about administering ASE, inline ASE, real-time cybersecurity and so on.
- ***ABS Admin Guide*** - Refer to the ABS admin guide to learn about administering ABS, AI engine training, various REST API reports and so on.

- ***Dashboard* on page 21** - Refer to the Dashboard admin guide to learn about how to access and use Dashboard.

Server requirements

The set up requires one machine which hosts all the six Docker images. The server requirement for the machine is specified in the following table:

OS	Ubuntu 16.04
Hardware	8 CPUs, 32 GB RAM, 500 GB Storage

ⓘ **Note:** The server requirement is for a single server for evaluation purpose only.

Docker version

The setup requires the Community Version (CE) of Docker 17.06 or higher. Make sure that the Docker infrastructure is set up before proceeding with installation and setup of PingIntelligence for APIs software.

Download and untar Docker package

Contact Ping Sales for instructions on accessing the Docker package. Download the Docker package and save it in the /opt directory.

Complete the following steps before Installing and loading the Docker images:

1. Untar the package by running the following command:

   ```
   $sudo tar -xf /opt/pi-api-docker-poc-4.2.tar.gz
   ```
2. Change the directory to `/opt/pingidentity/docker-poc`.

ⓘ **Note:** By default the Docker PoC package is configured to be deployed in inline mode. If you want to deploy Docker in sideband mode, see *Configure Docker PoC for sideband* on page 13.

Install ASE and ABS license
PingIntelligence ASE and ABS require a valid license to start. The license file for both the products is named `PingIntelligence.lic`. Complete the following

- **ASE:**

  Copy the ASE license file in the `pingidentity/docker-poc/license/ase` directory. Make sure that the license file is named as `PingIntelligence.lic` Following is a sample of the ASE license file:

  ```
  ID=981894
  Product=PingIntelligence
  Module=ASE
  Version=4.1
  IssueDate=2020-01-01
  EnforcementType=0
  ExpirationDate=2020-12-30
  Tier=Subscription
  SignCode=
  ```

```
Signature=
```

Verify that the correct file has been copied: To verify that the correct license file has been copied in the `pingidentity/docker-poc/license/ase` directory, run the following command:

```
# grep 'Module' license/ase/PingIntelligence.lic
Module=ASE
```

- **ABS:**

  Copy the ABS license file in the `pingidentity/docker-poc/license/abs` directory. Make sure that the license file is named as `PingIntelligence.lic`. Following is a sample of the ABS license file:

```
ID=981888
Product=PingIntelligence
Module=ABS
Version=4.1
IssueDate=2020-01-01
EnforcementType=0
ExpirationDate=2020-12-30
Tier=Subscription
SignCode=
Signature=
```

  Verify that the correct file has been copied: To verify that the correct license file has been copied in the `pingidentity/docker-poc/license/abs` directory, run the following command:

```
# grep 'Module' license/ase/PingIntelligence.lic
Module=ABS
```

Configure Docker PoC for sideband

You can optionally configure the Docker PoC environment for a sideband deployment with an API Gateway. The Docker PoC package ships with sample API swagger definition files which can be adapted to support your API Gateway environment. PingIntelligence sideband policies and documentation can be downloaded from the Ping download site.

Configure Docker package for sideband

Navigate to `config` directory and edit the `poc.config` file to set `mode` as `sideband`. Following is a sample `poc.config` file.

```
# 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 variables.

Variable	Description
ase_mode	Defines the deployment mode of ASE. Possible values are `inline` and `sideband`. Default value is `inline`.
training_period	Training period of AI engine in hours. Minimum value is 1-hour.
poc_mode	Defines the mode in which ABS AI engine trains its models. Default value is `true`. It is recommended to keep the value as `true`. If you change it to `false`, it may take longer time to set all the attack thresholds.
gateway_ip	Configure the URL for API gateway.
gateway_port	Port number of API gateway URL
gateway_protocol	API gateway protocol. Possible values are `ssl` or `tcp`.

Install and load Docker images

To install and load Docker images, enter the command on the host Ubuntu 16.04 machine. This command loads and installs the Docker images from the images directory:

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

```
root@ip-172-31-25-146:/opt/pingidentity/docker-poc# ./bin/start.sh install
Tue Mar 31 05:14:24 UTC 2020 : loading ASE image
Loaded image: pingidentity/ase:4.1.1
Tue Mar 31 05:14:28 UTC 2020 : loading ABS image
Loaded image: pingidentity/abs:4.1.1
Tue Mar 31 05:14:41 UTC 2020 : loading dashboard image
Loaded image: pingidentity/dashboard:4.1.1
Tue Mar 31 05:15:09 UTC 2020 : loading mongo image
Loaded image: pingidentity/mongo:4.2.0
Tue Mar 31 05:15:15 UTC 2020 : loading client image
Loaded image: pingidentity/client:4.1.1
Tue Mar 31 05:15:23 UTC 2020 : loading server image
Loaded image: pingidentity/server:4.1.1
Tue Mar 31 05:15:28 UTC 2020 : Installation completed successfully
```

Setup the PoC environment

To start the Docker containers and setup, enter the following command the on the host Ubuntu 16.04 machine:

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

```
Tue Mar 31 05:12:28 UTC 2020 : Starting setup scripts
```

```
vm.max_map_count = 262144
Training period configured: 1 hour(s)
Creating network pingidentity_net
Creating service pingidentity_ase
Creating service pingidentity_dashboard
Creating service pingidentity_server
Creating service pingidentity_client
Creating service pingidentity_mongo
Creating service pingidentity_abs
Tue Mar 31 05:12:30 UTC 2020 : Setup successful
```

Verify ASE and ABS startup

Wait for a minute after the successful completion of the set up and enter the following command to verify that ASE and ABS have started:

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

If a wrong license was installed, the following error is displayed:

```
/opt/pingidentity/docker-poc#sudo ./bin/start.sh setup
Tue Dec  31 05:12:45 UTC 2019 : Starting setup scripts
Creating network pingidentity_net
open /opt/pingidentity/docker-poc/license/ase/PingIntelligence.lic: no such file or directory
Tue Dec  31 05:12:46 UTC 2019 : Error : Error during setup
```

Start the training

The PingIntelligence for APIs AI engine needs to be trained before it can start detecting attacks on your APIs. Enter the following command to start the training. The training duration is 85 minutes.

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

```
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.
```

Generate sample attacks

To generate sample attacks on the preconfigured APIs, enter the following command:

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

```
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.
```

API deception

You can view the deception APIs by running the following command. The deception API is part of the set up. The deception command completes the following steps:

- Enables ASE detected attacks
- Fetches the list of configured APIs from ASE
- Sends traffic to the decoy API and receives a 200 OK response

- Send traffic to a regular API (for example, shopapi). The connection is blocked because any client which previously accessed a decoy API is not allowed access to “production” APIs.

> ⓘ **Note:** API deception works only for inline Docker PoC setup.

Execute the following script to test API deception:

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

API discovery

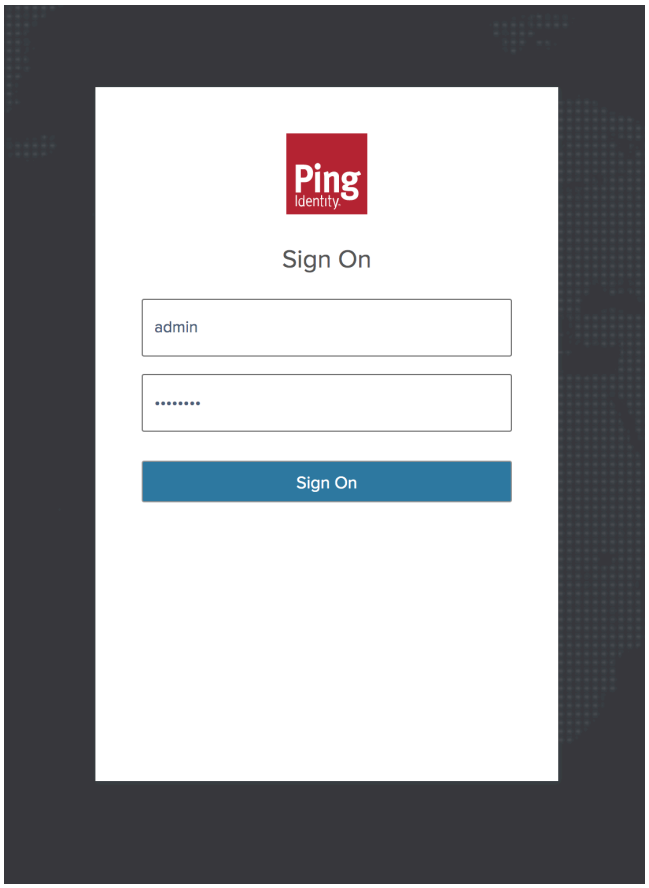
Automated API Definition (AAD) tool is installed as part of the setup. ABS discovers the APIs when the discovery is enabled. The automated setup sets up the discovery mode. APIs are discovered by ABS when a global API is defined in PingIntelligence ASE. AAD fetches the discovered APIs from ABS and adds them in ASE. API model training starts after the APIs are added in ASE. For more information, See *API discovery and configuration* on page 321.

Access PingIntelligence Dashboard

Access the PingIntelligence for APIs Dashboard from a browser at this default URL: `https://<pi_install_host>:8030`.

At the login screen, login 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 PoC deployment use the default password.



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

PingIntelligence Dashboard is categorized into the following components:

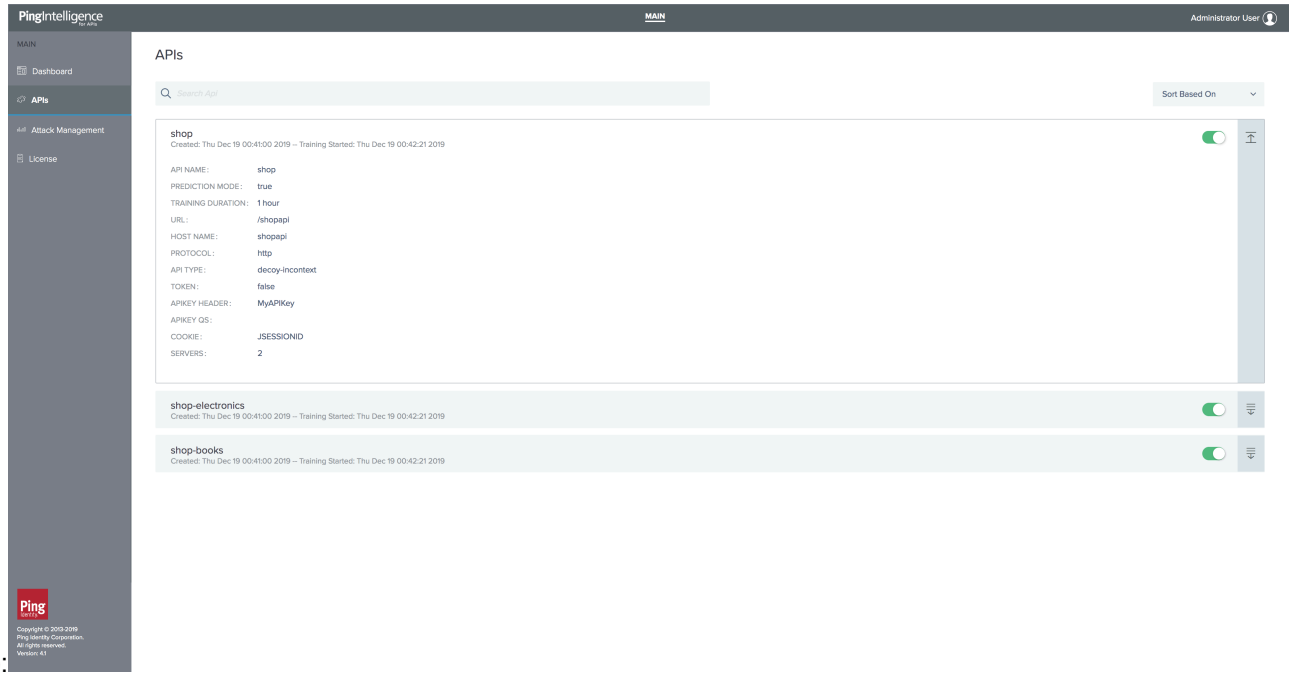
- **Main Dashboard** - Available for `admin` and `ping_user`
- **APIs** - Available only for `admin` user
- **Discovered APIs** - Available only for `admin` user
- **Attack Management** - Available only for `admin` user
- **License**

APIs

The API tab displays all the APIs available in ABS AI engine and displays the following information:

- **API name**: API name used by PingIntelligence
- **Prediction mode**: A `true` status means that at least one system generated threshold value is set, while a `false` status means that the API is still under training mode
- **Training duration**: The minimum configured time in hours configured in ABS AI engine to train an API. This is configured in `abs_init.js` in ABS. For more information, see *AI Engine training* on page 312
- **URL**: API basepath URL configured in the API JSON file. For more information, see *API JSON definition*
- **Host name**: Host name of the API configured in the API JSON file. For more information, see *API JSON definition*
- **Protocol**: The protocol configured in the API JSON file. For more information, see *API JSON definition*
- **API type**: API type can be `regular`, `decoy - incontext`, or `decoy-out-of-context`. For more information on deception, see *API deception environment* on page 218
- **Token**: A `true` status means that PingIntelligence will use OAuth tokens for reporting and attack detection. For more information, see *API JSON definition*
- **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 *API JSON definition*
- **Cookie**: The cookie value configured in the API JSON file and used for reporting and attack detection. Displays blank, if cookie was not configured in API JSON. For more information, see *API JSON definition*
- **Servers**: The backend API server configured in the API JSON file - "*" supports all the host names. For more information, see *API JSON definition*

Using the toggle button, you can 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 keeps fetching API data from ABS AI engine. 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. Here is a screenshot of APIs tab:

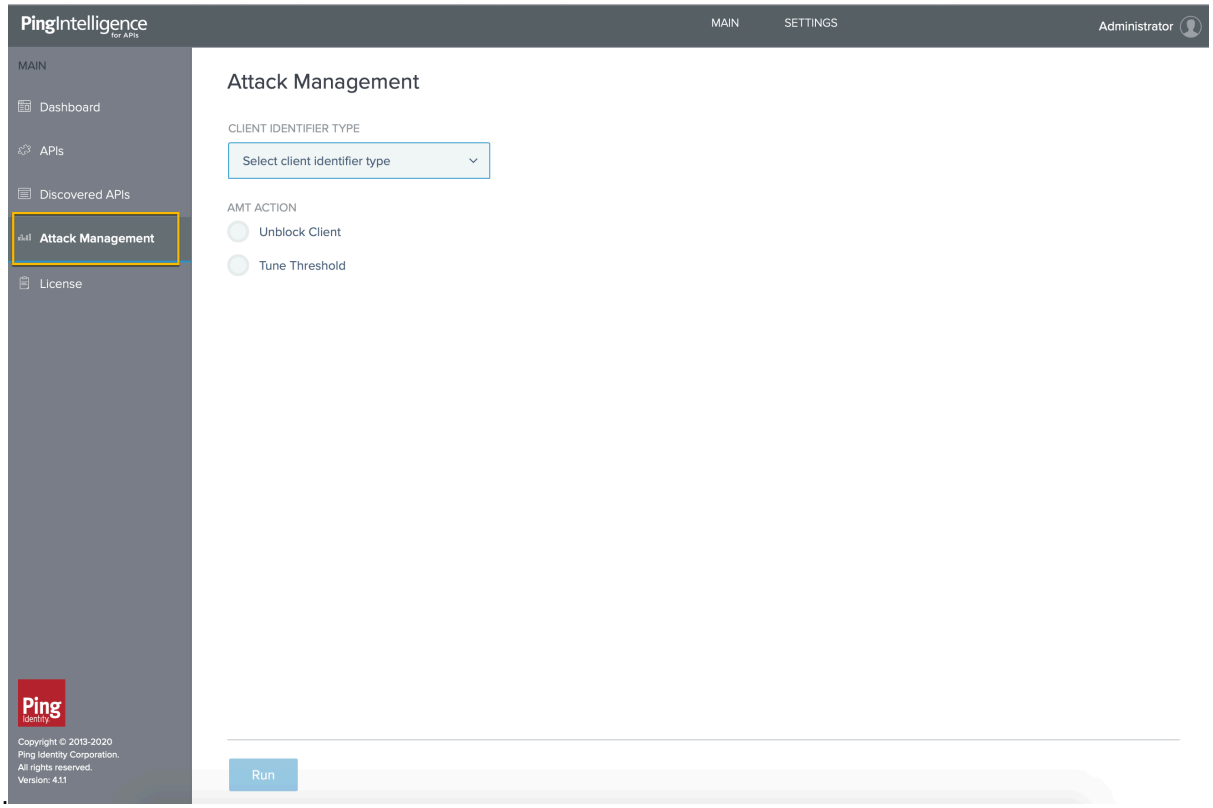


Attack management

The attack management feature of PingIntelligence for APIs Dashboard supports unblocking of clients and tuning thresholds values for attacks. Click on the **Attack Management** tab on the left pane to access it.

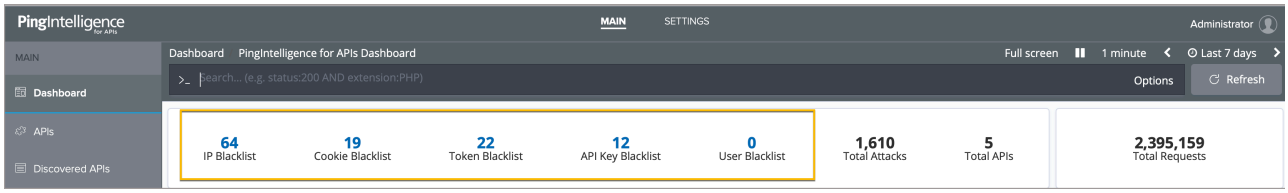
> ⓘ **Note:** The Attack management feature is available only for an Admin user. You need to have Admin user privileges to perform **Unblock** and **Tune** operations on a client identifier.

The following screenshot illustrates the Attack Management UI.



Interactive blacklists

The PingIntelligence for APIs Dashboard provides the capability of unblocking or tuning a blacklist directly from the Dashboard. The user can select the client identifier and the Attack management action from the Dashboard. For more information, see *Interactive blacklists* on page 22. The following screen shot shows the client identifier blacklists across APIs in the Dashboard.



> ⓘ **Note:** When the user initiates Attack management from the Dashboard, the values for the client identifiers are auto-populated except the API key key-name.

Unblock a client identifier

Complete the following steps to unblock a client identifier:

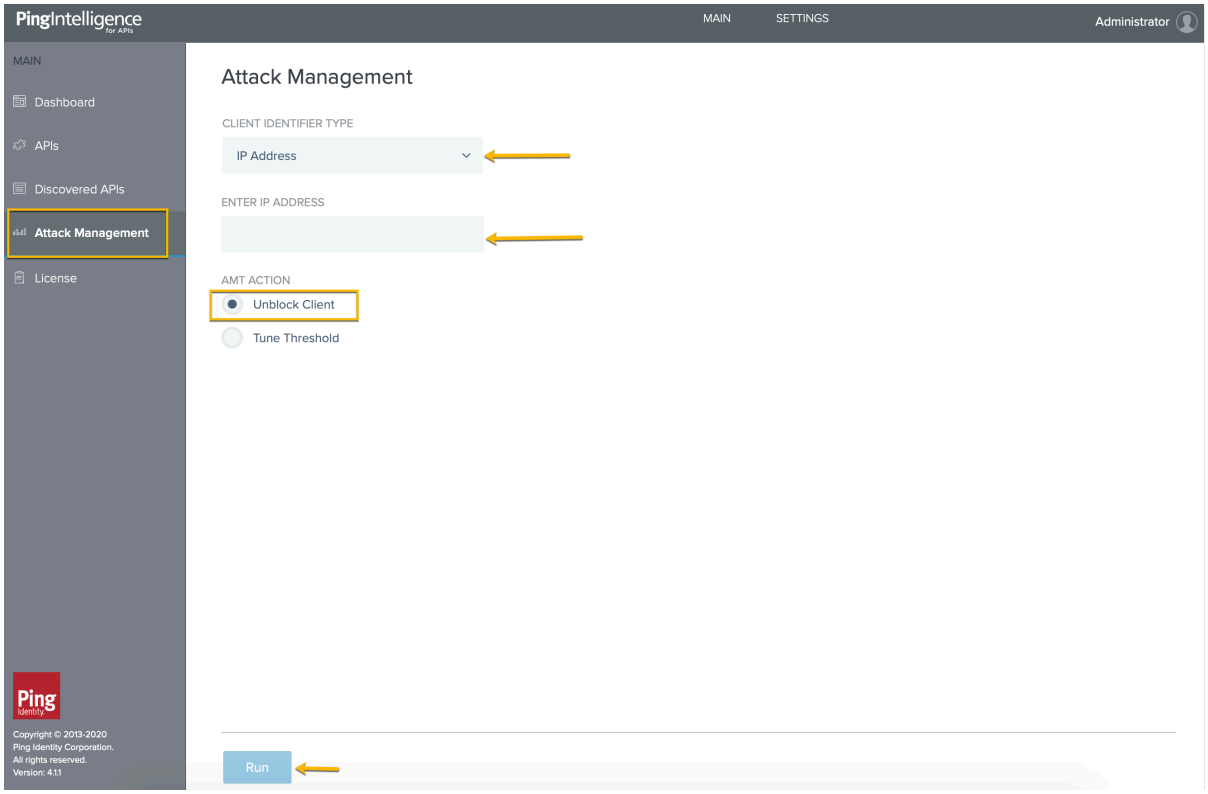
1. Select the type of client identifier from the **Client Identifier Type** list.

2. Enter the value of the client identifier.

> **Note:** For API Key and Cookie, enter the name and the value.

3. Select the **Unblock Client** check box.
4. Click **Run**.

The following screen shot illustrates the unblock client operation.



The unblock operation deletes the client identifier from the PingIntelligence ASE and ABS AI engine blacklist. To verify that the client identifier has been deleted from ASE, run the **`view_blacklist`** CLI command or blacklist REST API in ASE. To verify that the client identifier has been deleted from ABS, use the `attacklist` REST API. For more information on ABS blacklist, see *ABS blacklist reporting* on page 342.

> **Note:** The API keys will not be deleted from the blacklist immediately in ASE if the API Key key-name is not entered. The deletion is delayed until ASE retrieves the blacklist data from ABS.

Tune threshold

To address false positives, the **Attack Management** feature supports automatic threshold tuning. When tuning thresholds for a specific client identifier, the Attack management functionality does the following:

1. It fetches all the attacks flagged for the client identifier from ABS AI Engine.
2. After it has identified all the attacks, it increases the threshold values for those attacks. At this point, the threshold has moved from `system` defined to `user` defined. For more information on thresholds, see *Tune thresholds for false positives* on page 317.

Complete the following steps to tune thresholds:

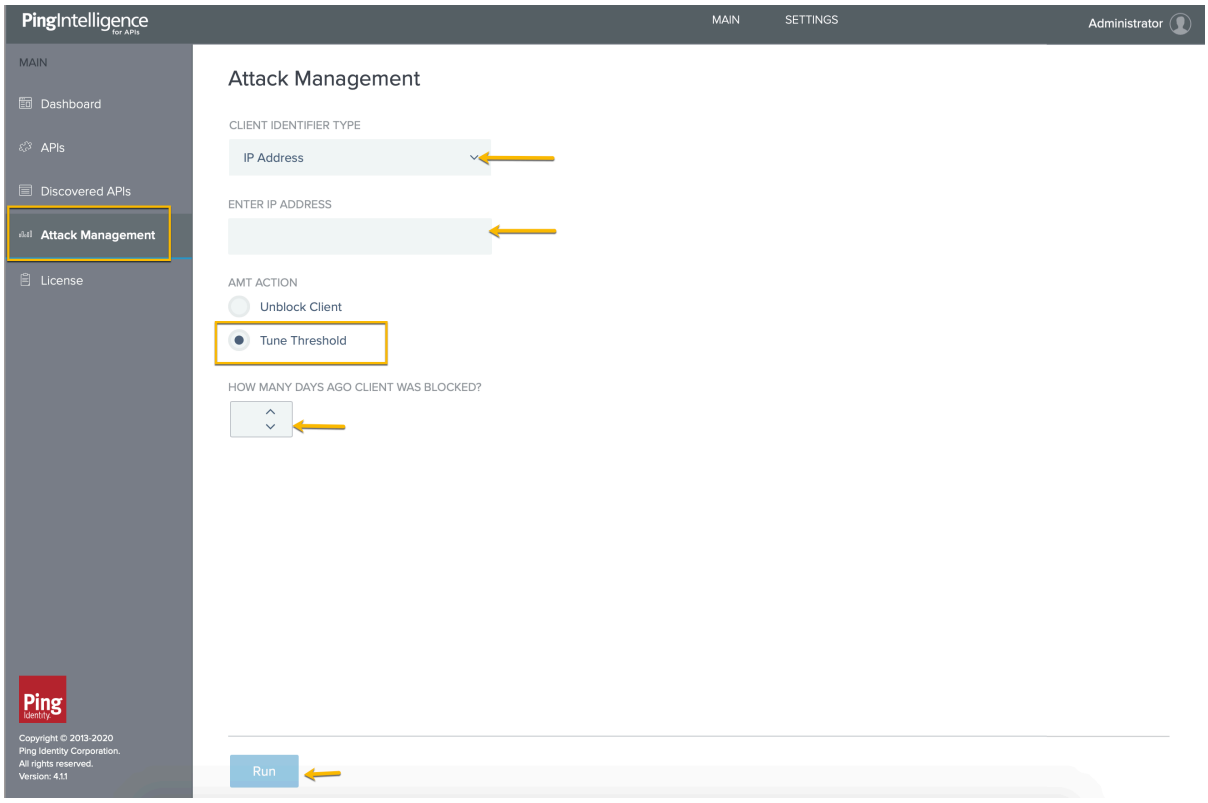
1. Select the type of client identifier from the **Client Identifier Type** list.
2. Enter the value of the client identifier.
3. Select the **Tune Threshold** check box.

4. Provide the approximate number of days since the client was blocked. The maximum value is 30-days.

> ⓘ **Note:** The value for **How many days ago client was blocked?** gets auto-populated when Attack Management is initiated from the Dashboard interactive blacklist. The value is calculated as follows,
>
> ```
> How many days ago client was blocked? = Current date - Attack detection
>  date + 1
> ```
>
> When auto-populating, if the calculated value is more than 30 days, it is trimmed down to 30.You can use the same formula when populating the value manually. The Attack detection date for a client identifier is available in the interactive blacklists.

5. Click **Run**.

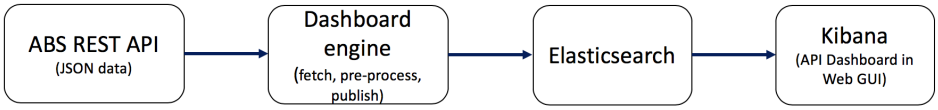
The following screen shot illustrates tuning threshold for a client identifier.



Dashboard

The Dashboard provides a near real-time snapshot of your API environment. It provides insights on user activity, attack information, blocked connections, forensic data, and much more. The Dashboard makes periodic REST API calls to the ABS (API Behavioral Security) AI engine, which returns JSON reports that

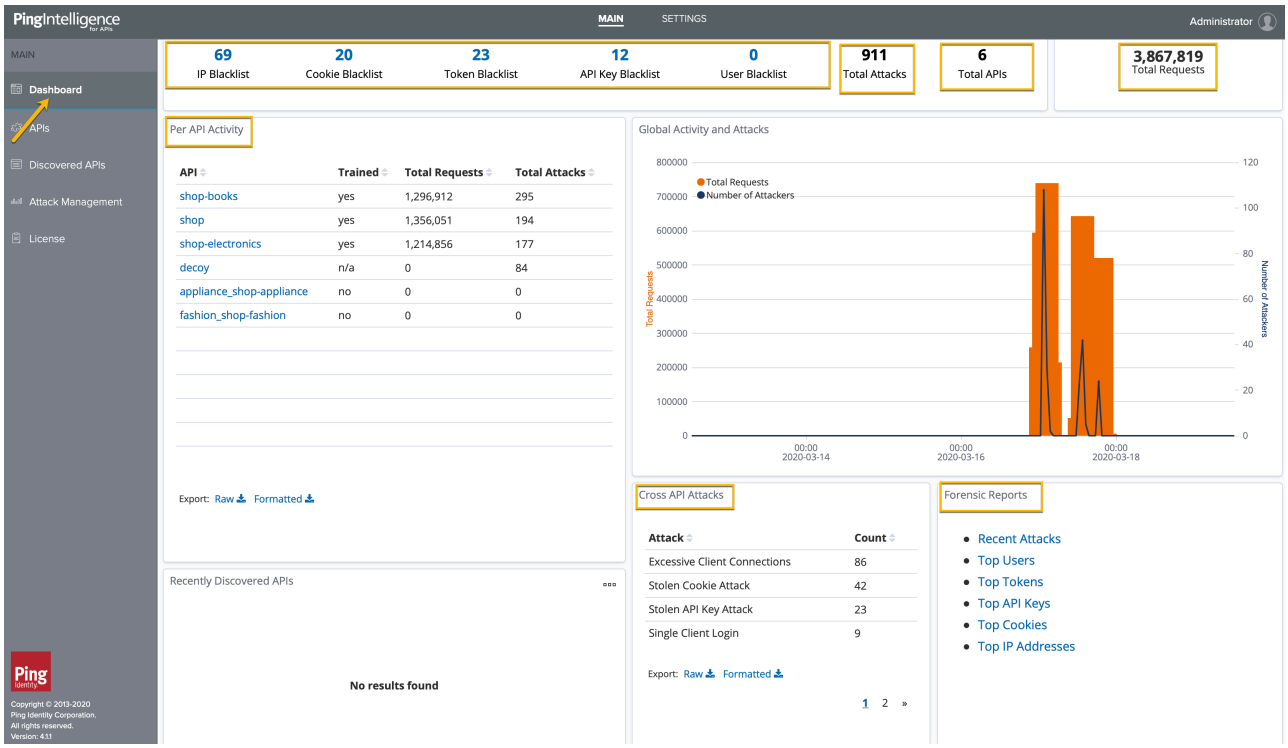
are used to generate visualizations and API metrics. The following illustration shows the data flow for API



dashboard.

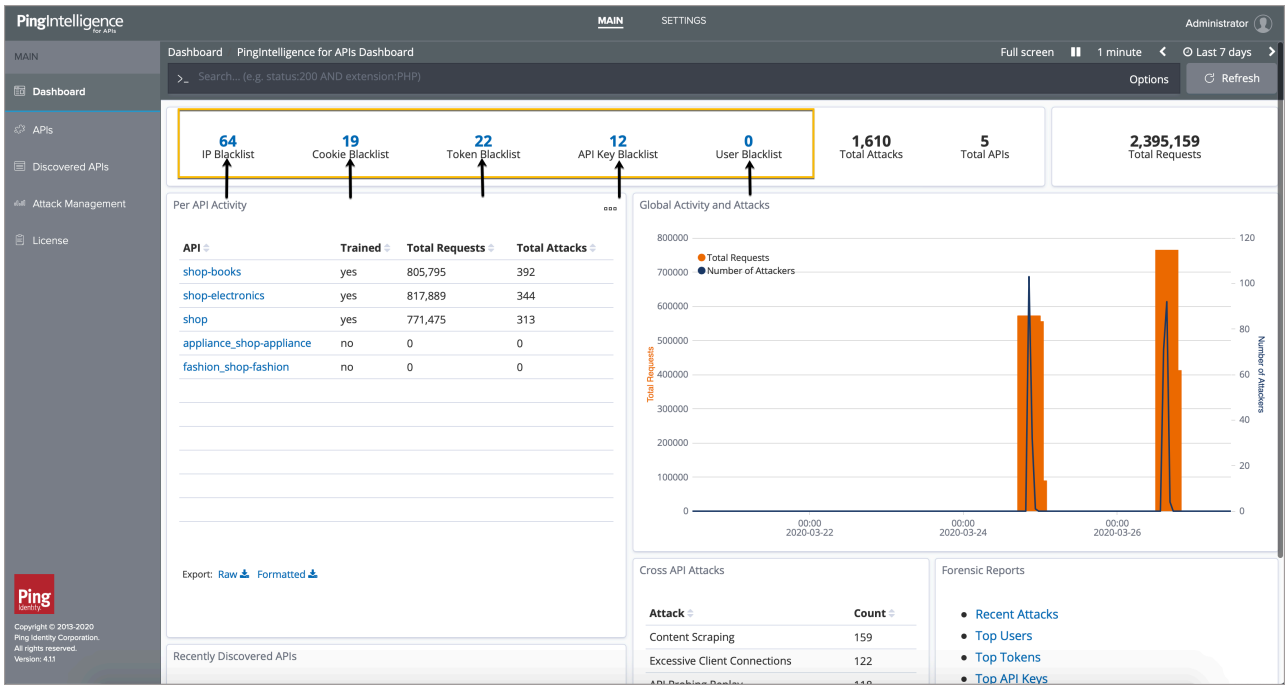
To view the API dashboard, click on **Dashboard**. The Dashboard provides information on the following::

- Global metrics like:
  - Blacklist across APIs for each client identifier. For more information, see *Interactive blacklists* on page 22.
  - Total attacks across APIs
  - Total requests across APIs
  - Number of APIs in your environment
- Time series visualization of total number of requests and attacks. For more information, see *Dashboard time series* on page 24.
- Data on Per API activity. For more information, see *Per API activity* on page 457.
- Data on attacks across APIs. For more information, see *Cross API attacks and recently discovered APIs* on page 464.
- Forensic reports across APIs. For more information, see *Forensic reports* on page 459.
- Recently discovered APIs in the environment.



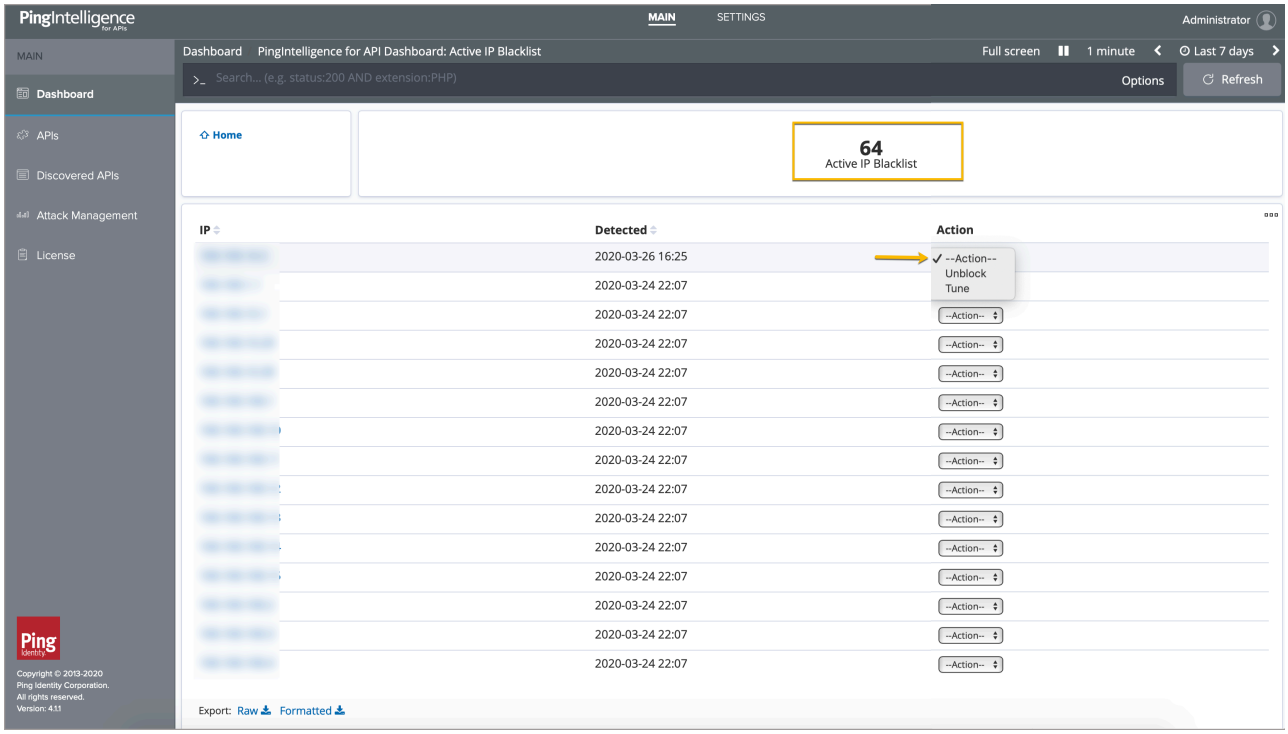
Interactive blacklists

PingIntelligence for APIs Dashboard provides the capability of interactive blacklist management. A blacklist is a list of client identifiers that were detected executing an attack. The dashboard enables you to unblock the blacklisted client identifiers or tune the threshold values for attack types. It supports the following client identifier types- IP address, Cookie, Token, API Key, and Username. You can view the top-500 entries on each blacklist from the dashboard.



Click on the count for any blacklist type, for example, **IP Blacklist**. The dashboard lists the blacklisted IP addresses along with the Detected date..

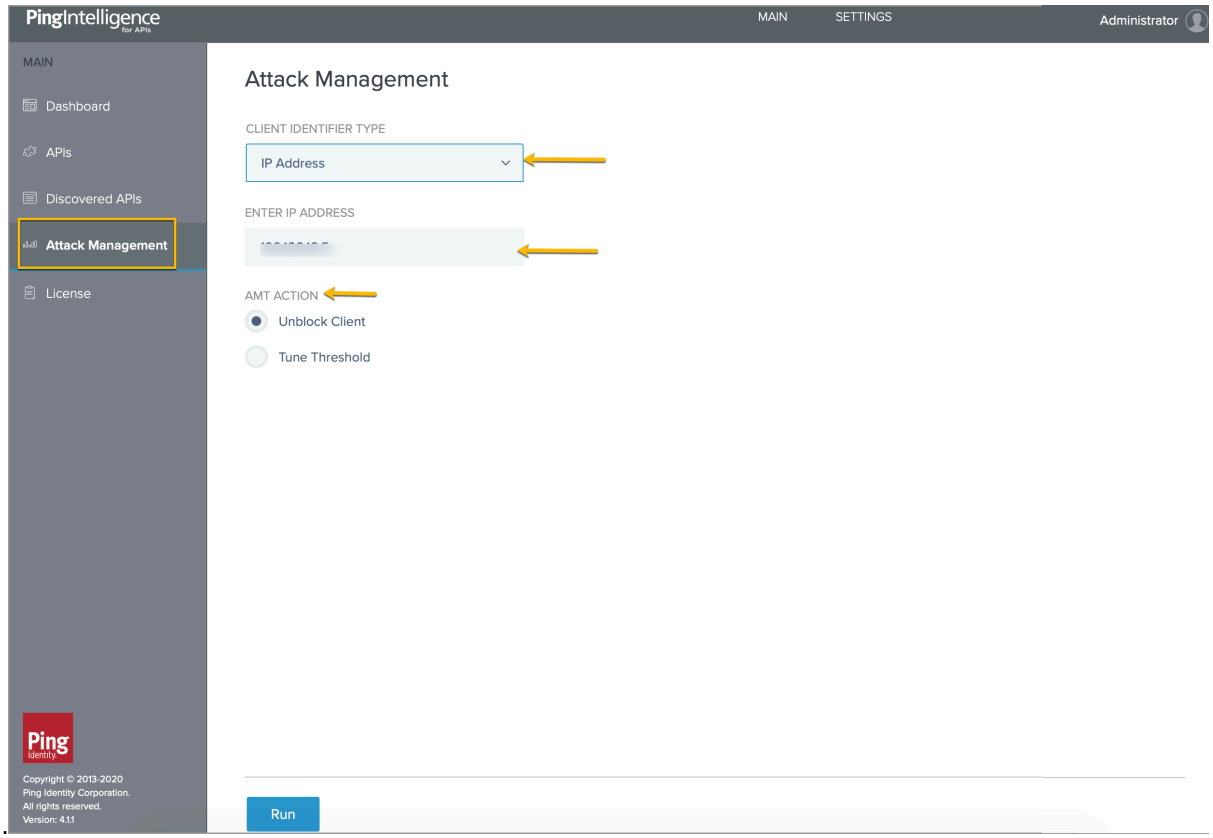
The following screenshot shows the expanded blacklist:



For each blacklisted IP address, you get the option to Unblock or Tune in the **Action** list. Clicking on either action redirects the dashboard to the Attack management application. Attack management allows you to run the operations for unblocking the client identifiers and tuning the threshold values.

ⓘ **Note:** The **Action** list is available only for an Admin user. You need to have Admin user privileges to perform**Unblock** and **Tune** operations on a client identifier.

The following screen shot shows the Attack management UI.



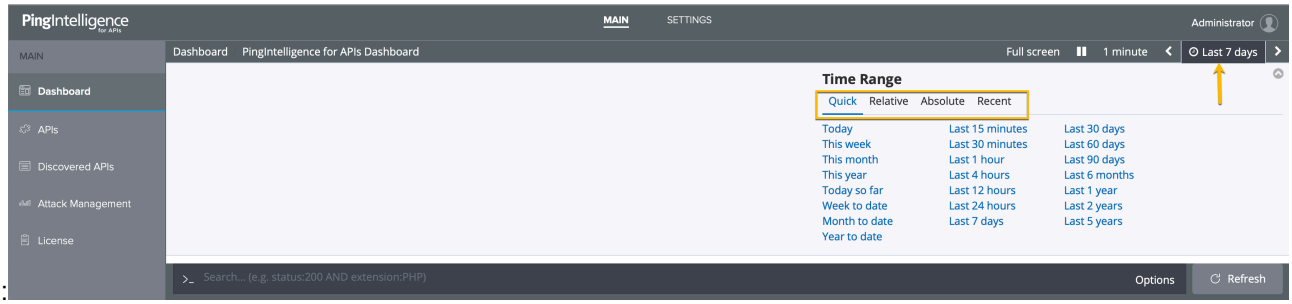
The values in **Client Identifier Type** and **Enter IP Address** get auto-populated into the Attack management application from the dashboard. The **AMT Action** is auto-selected. Click **Run** to execute the operation. For more information on Attack management, see *Attack management* on page 19.

> ⓘ **Note:** Dashboard does not populate the API key key-name in the Attack management application when the client identifier is API key. It only populates the API key value.

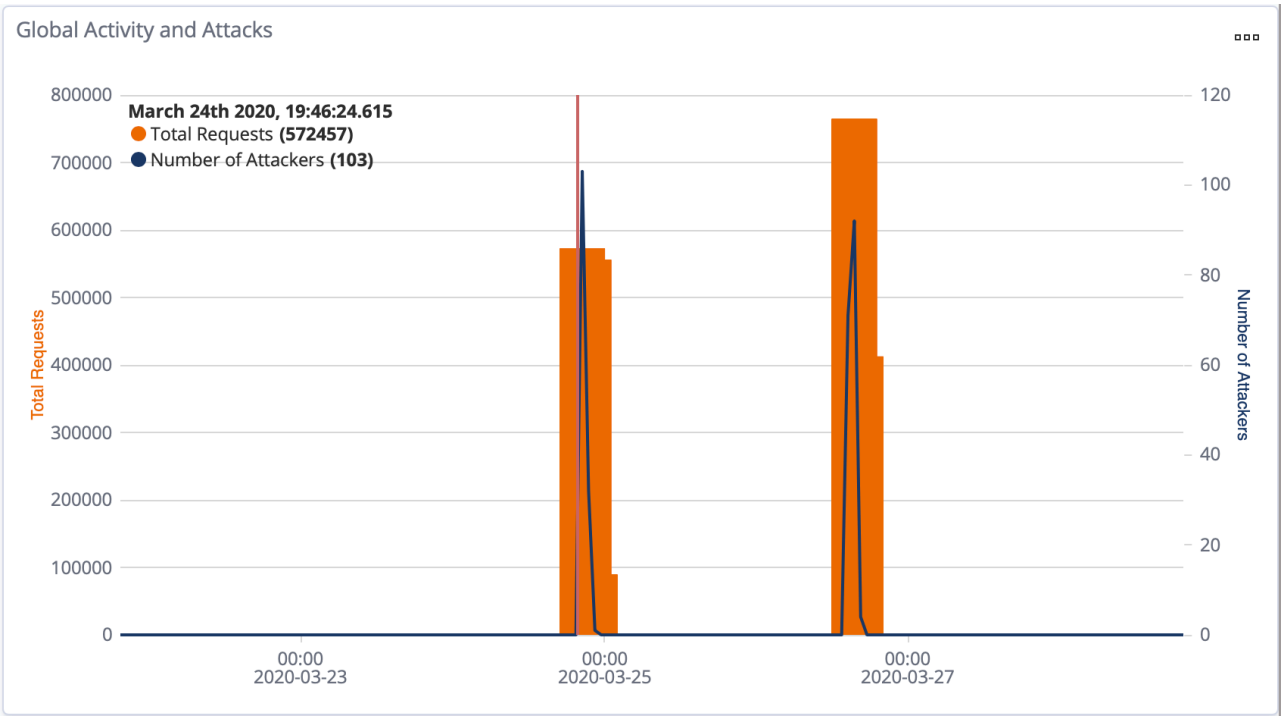
Dashboard time series

PingIntelligence Dashboard shows the attacks in a time-series format. To adjust the timeframe viewed on the Dashboard, click between the **time-period** arrows located on the top right corner of the dashboard and select the desired time period.

See the example in the following screen capture:



The following screen capture shows the total requests and number of attackers data in time series format.

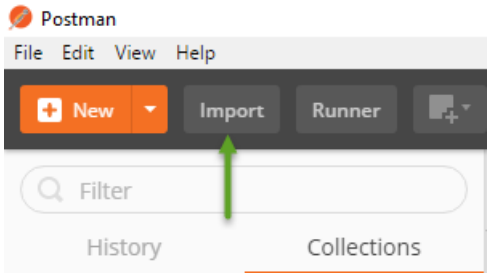


ABS detailed reporting

ABS Engine's REST API interface provides access to a range of JSON reports on attacks, metrics, and anomalies. To view these reports, Ping Identity provides templates which can be loaded into Postman to simplify viewing of the JSON reports.

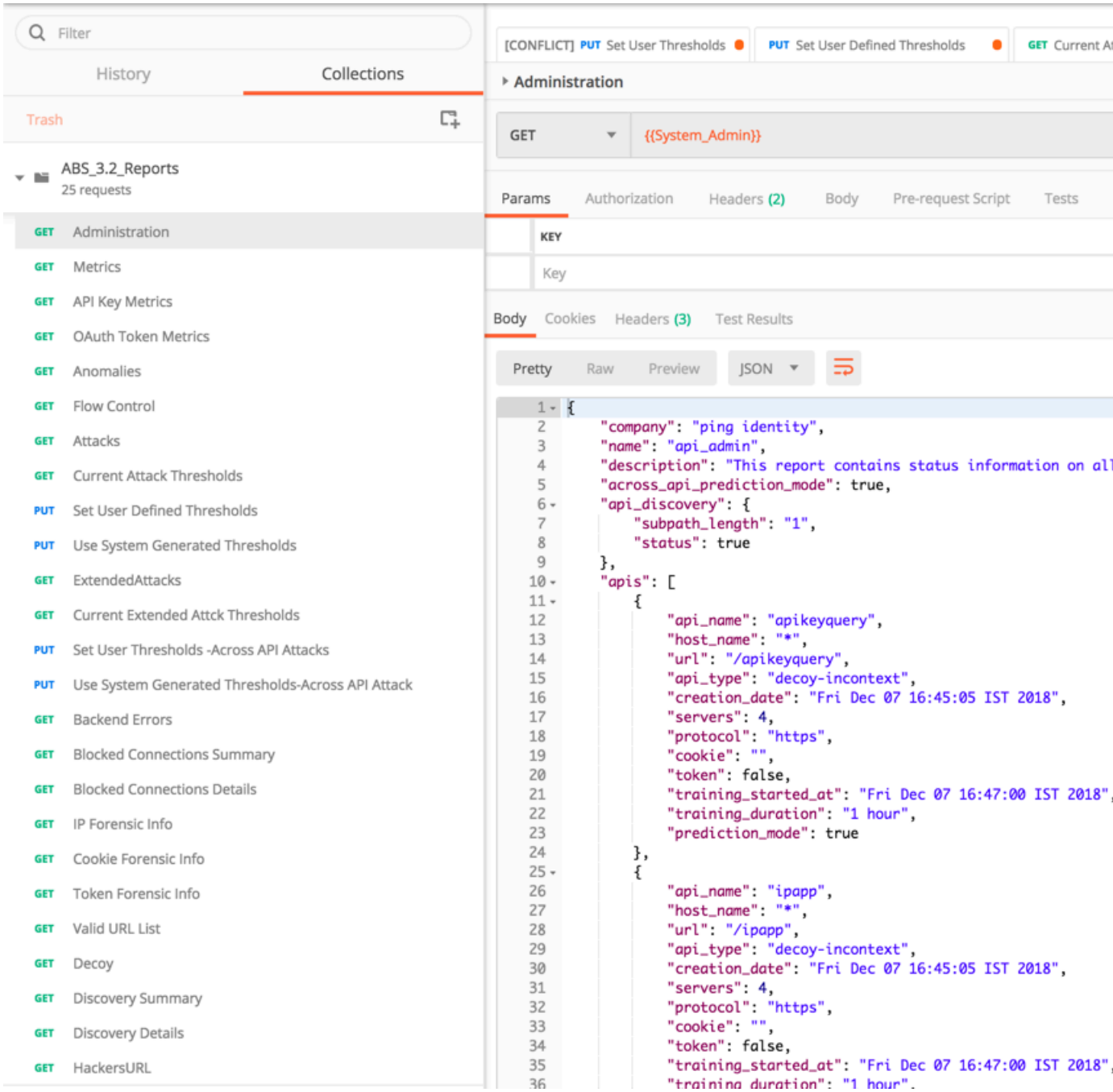
Install and Configure Postman Software

1. *Download* and install the Postman application 6.2.5 or higher.
2. *Download* "API Reports Using Postman Collection" from the Automated Docker PoC Installation section of the download site. `ABS_4.1_Environment` and `ABS_4.1_Reports` are files for Postman.
3. Launch the Postman application. Make sure to disable SSL verification in Postman. For more information, see *Using self-signed certificate with Postman*
4. Import the downloaded reports files by clicking the **Import** button



5. Click the gear  button in the top right corner
6. In the pop-up window, click **ABS_4.2_Environment.**

7. In the Edit Environment pop-up window, configure the following values and click **Update**.
   a. Server IP Address – IP address of the Docker machine
   b. Port – Default is 8080
   c. Access_Key, Secret_Key - Default Access_Key is `abs_ak` and default Secret_Key is `abs_sk`
   d. API_Name – the name of API to view in reports
   e. 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**.



Other reports which can be generated for a specified time-frame (make sure you specify a time range which covers the time that you ran the attack scripts) include:

- Metrics – shows all activity on the specified API

- 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 which 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

Shutdown the PoC environment

You can stop the Docker PoC setup by entering the following command to delete all containers and the data.

```
root@vortex-108:/opt/pingidentity/docker-inline-poc$sudo ./bin/stop.sh
Tue Dec 31 09:13:02 UTC 2019 : Starting stop scripts
Removing service pingidentity_aad
Removing service pingidentity_abs
Removing service pingidentity_ase
Removing service pingidentity_client
Removing service pingidentity_mongo
Removing service pingidentity_server
Removing service pingidentity_webgui
Removing network pingidentity_net
Tue Dec 31 09:14:03 UTC 2019 : Stop successful
```

Appendix: Verify the Setup

Carry out the following basic steps to verify the setup:

1. **Listing the Docker Containers**

   List all the containers with the `docker ps` command.
2. **Get Console Access:**

   To get console access for any of the Docker, fetch the Container ID of the Docker using the `docker ps` command output and use it in the following command:

   `#docker exec -it <docker-container-id> /bin/bash`
3. **PingIntelligence for APIs Products:**

   The Intelligence products are installed in the `/opt/pingidentity` directory within the Docker.
4. **Checking the service names:**

   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.
5. **Checking the logs of service:**

   To check the log of any service, use the following command:

   `#docker service logs <service name>`

   For example `docker service logs pingidentity_ase`

PingIntelligence Kubernetes PoC deployment

PingIntelligence Kubernetes PoC deployment

PingIntelligence ships an example `yml` file with its Docker toolkit package. You can use this example `yml` file to deploy PingIntelligence on a Kubernetes cluster node. This document describes installing PingIntelligence on an on-premise Kubernetes cluster node. The setup uses Minikube for PingIntelligence deployment.

The example `yml` file creates the following resources in the Kubernetes cluster:

- **4 statefulsets** with one container each for MongoDB, ABS AI engine, ASE, and Dashboard.
- **5 external services (NodePort type)** - Two each for ABS AI engine and ASE and one for the Dashboard.
- **3 internal services (clusterIP type)** - One each for MongoDB, ABS AI engine and ASE.

Prerequisites

PingIntelligence PoC deployment on Kubernetes cluster node has the following prerequisites:

- A virtual machine or a bare metal server with 8 CPUs, 32 GB of RAM, and 500 GB of hard disk.
- Docker engine - version 19.03.7 on Ubuntu or version 1.13 on RHEL. If you want a native Kubernetes cluster installation, then Minikube requires a pre-installed docker engine.
- Minikube version 1.7.3
- Kubectl CLI version 1.6.0 to interact with Kubernetes cluster.

Deployment overview

Deploying PingIntelligence in a Kubernetes cluster consists of the following steps:

1. Install Docker on RHEL or Ubuntu host
2. Install Minikube
3. Download and install kubectl
4. Creating a single node Kubernetes cluster with Minikube
5. Deploy PingIntelligence in Kubernetes cluster

> ⓘ **Note:** This deployment of PingIntelligence on a Kubernetes cluster node is suitable for POC environments only. It is not suitable for production environments or for security testing environments.

Install Docker on RHEL or Ubuntu

About this task

The following steps describe installing Docker engine on Ubuntu and RHEL

Steps

1. Install `docker-ce` on an RHEL or Ubuntu host by entering the commands show below.

   RHEL

```
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
$ sudo yum-config-manager --enable rhel-7-server-extras-rpms
```

```
$ sudo yum install docker
```

Ubuntu

```
$ sudo apt update
$ sudo apt-get install docker-ce=19.03.7
```

2. Start docker by entering the following command

```
$ sudo systemctl start docker
```

Install minikube and kubectl

About this task

Complete the following steps to install minikube and kubectl on your host machine:

Steps

1. Install minikube on RHEL or Ubuntu host.

   RHEL 7.6

   ```
   $ sudo yum install -y https://storage.googleapis.com/minikube/releases/
   latest/minikube-1.7.3-0.x86_64.rpm
   ```

   Ubuntu

   ```
   $ curl -LO https://storage.googleapis.com/minikube/releases/latest/
   minikube_1.7.3-0_amd64.deb && sudo dpkg -i minikube_1.7.3-0_amd64.deb
   ```

2. Download kubectl

   ```
   $ curl -LO https://storage.googleapis.com/kubernetes-release/release/
   v1.16.0/bin/linux/amd64/kubectl && chmod +x ./kubectl
   ```

3. Install kubectl

   ```
   $ sudo install kubectl /usr/bin/
   ```

Install Kubernetes cluster node

About this task

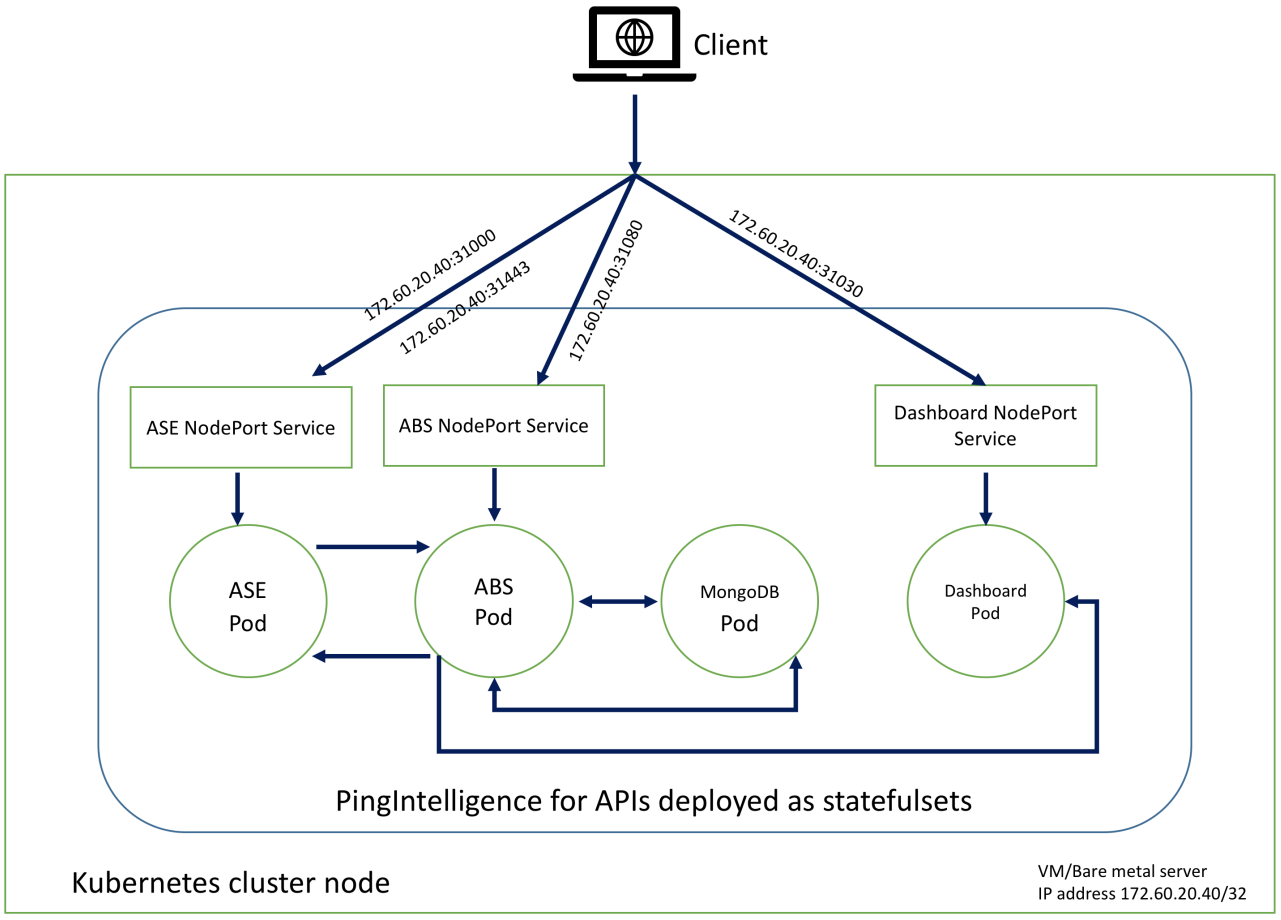
Complete the following step to install Kubernetes cluster node:

Steps

Run the following command

```
$ sudo minikube start --kubernetes-version v1.16.0 --vm-driver=none
```

The following diagram shows the PingIntelligence deployment in a Kubernetes cluster



Deploy PingIntelligence in Kubernetes cluster

About this task

Complete the following steps to deploy PingIntelligence in a Kubernetes cluster:

Steps

1. Download PingIntelligence Docker toolkit from the *download* site.
2. Untar the docker toolkit by entering the following command.

```
tar -zxf pi-api-docker-toolkit-4.1.1.tar.gz
```

3. Build the PingIntelligence docker images by completing the steps mentioned in *Build the PingIntelligence Docker images* on page 679 topic.
4. Navigate to `pingidentity/docker-toolkit/examples/kubernetes` directory to edit the `pi4api-k8s-poc` file.
5. Edit the environment variable in `pi4api-k8s-poc` file to configure the ASE deployment mode. The values can be inline or sideband. Following is a snippet of the file showing the environment variable.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: ase
  labels:
```

```
    app: ase
spec:
  serviceName: ase-internal-service
  replicas: 1
  selector:
    matchLabels:
      app: ase
  template:
    metadata:
      labels:
        app: ase
    spec:
      terminationGracePeriodSeconds: 60
      securityContext:
        runAsUser: 10001
        fsGroup: 0
      containers:
        - name: ase
          image: pingidentity/ase:4.1.1
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080
              name: management
            - containerPort: 9090
              name: logs
          command:
            - "/bin/bash"
            - "-c"
            - "/opt/pingidentity/ase/entrypoint.sh"
          env:
            - name: TZ
              value: "Etc/UTC"
            - name: MODE
       value: "inline"

            - name: ENABLE_CLUSTER
              value: "true"
            - name: ENABLE_ABS
              value: "true"
            - name: ABS_ENDPOINT
              value: "abs-0.abs-internal-service:8080"
            - name: ABS_ACCESS_KEY
              value: "abs_ak"
            - name: ABS_SECRET_KEY
              value: "abs_sk"
```

6. Add ABS and ASE license in the ConfigMaps section of the `pi4api-k8s-poc` file.

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: abs-license
data:
  PingIntelligence.lic: |
    ID=
    Organization=
    Product=PingIntelligence
    Module=ABS
    Version=4.1
    IssueDate=
    EnforcementType=
    ExpirationDate=
```

```
    MaxTransactionsPerMonth=
    Tier=
    SignCode=
    Signature=

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ase-license
data:
  PingIntelligence.lic: |
    ID=
    Product=PingIntelligence
    Module=ASE
    Version=4.1
    IssueDate=
    EnforcementType=
    ExpirationDate=
    MaxTransactionsPerMonth=
    Tier=
    SignCode=
    Signature=
```

7. Create a namespace.

```
$ sudo su
# kubectl create namespace pingidentity
```

8. Apply the edited `pi4api-k8s-poc.yml` file to deploy the resources on the Kubernetes cluster.

```
# kubectl apply -f pi4api-k8s-poc.yml -n pingidentity

daemonset.apps/startup-script created
statefulset.apps/mongo created
statefulset.apps/abs created
statefulset.apps/ase created
statefulset.apps/dashboard created
service/abs-external-service created
service/ase-external-service created
service/dashboard-external-service created
service/mongo-internal-service created
service/abs-internal-service created
service/ase-internal-service created
```

Next steps
Verify that the deployment is successful by entering the following command.

```
# kubectl get pod -n pingidentity
NAME                  READY   STATUS    RESTARTS   AGE
abs-0                 1/1     Running   0          139m
ase-0                 1/1     Running   0          25m
mongo-0               1/1     Running   1          139m
startup-script-5d5d6  1/1     Running   0          119m
dashboard-0           1/1     Running   1          139m
```

Fetch the IP addresses of ASE, ABS, and Dashboard by entering the following command.

```
# kubectl get svc -n pingidentity
NAME            TYPE     CLUSTER-IP     EXTERNAL-IP   PORT(S)                AGE
```

```
abs-external-service      NodePort    10.100.81.119    <none>
 8080:31080/TCP,9090:31090/TCP   3m12s
abs-internal-service      ClusterIP   None             <none>        8080/
TCP                        3m12s
ase-external-service      NodePort    10.104.103.138   <none>
 80:31000/TCP,443:31443/TCP      3m12s
ase-internal-service      ClusterIP   None             <none>        8020/
TCP,8010/TCP               3m12s
mongo-internal-service    ClusterIP   None             <none>        27017/
TCP                       3m12s
dashboard-external-service   NodePort    10.100.8.48      <none>
 443:31030/TCP                3m12s
```

If you are deploying in the sideband mode, take the NodePort IP address of ASE to use in API gateway integration.

PingIntelligence Cloud service deployment

PingIntelligence Cloud service

PingIntelligence Cloud deployment has two components which work together to complete your PingIntelligence PoC environment. The PingIntelligence Cloud environment is distributed between the following:

- PingIntelligence ABS, Dashboard, and MongoDB are hosted as a cloud service managed by Ping Identity
- PingIntelligence Cloud Connector (referred to as PingIntelligence ASE in the documentation) is deployed in your API environment.

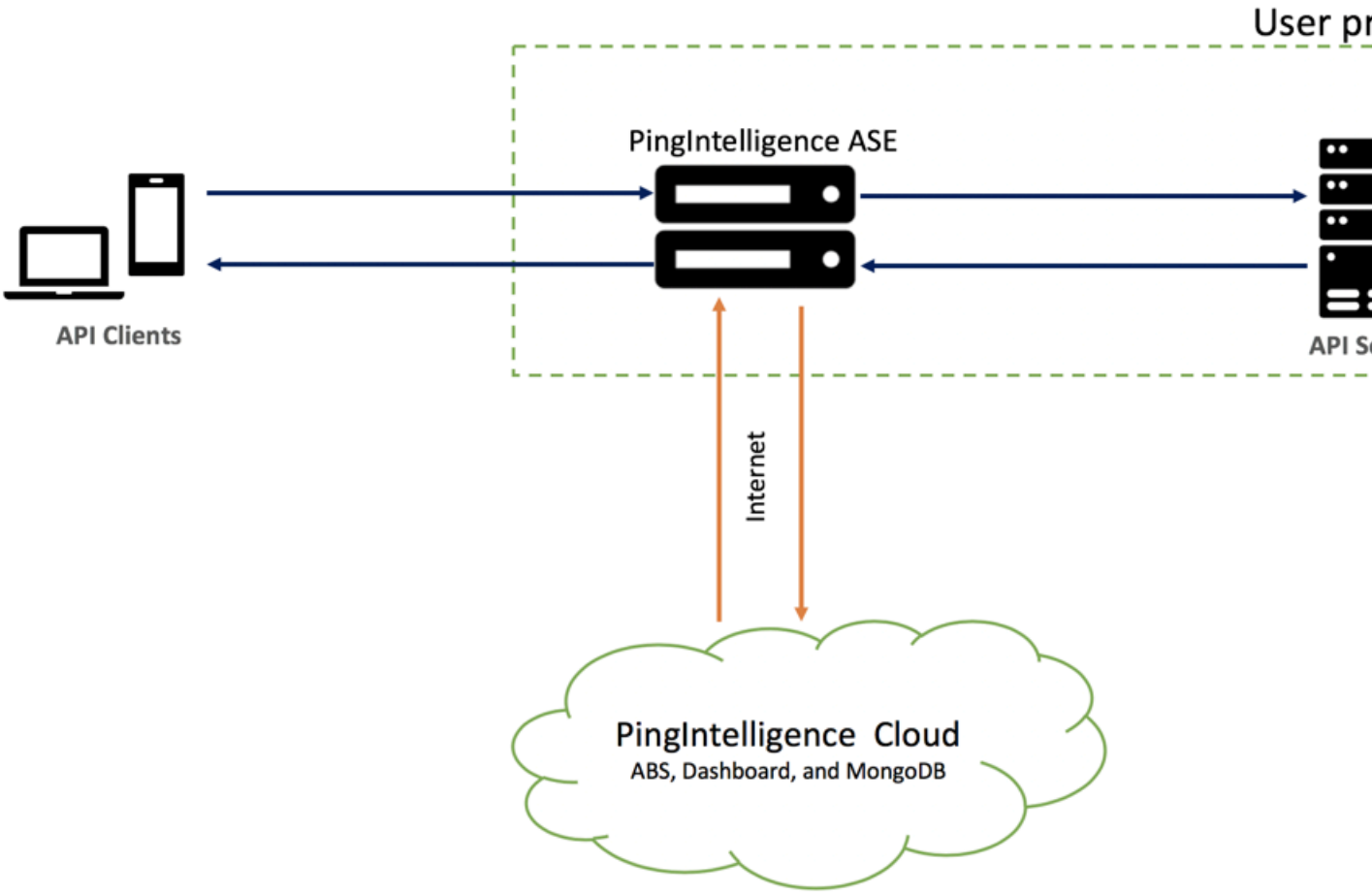
PingIntelligence Cloud service can be deployed in two modes:

- Inline mode
- Sideband mode

Inline mode

In inline mode, ASE receives API client traffic and routes the traffic to API servers. It can be deployed behind an existing load balancer, such as AWS ELB. In inline mode, ASE terminates SSL connections from API clients and then routers the API requests to the target APIs – running on an API Gateway or app servers, such as Node.js, WebLogic, Tomcat, PHP, etc. To configure ASE to work in Inline

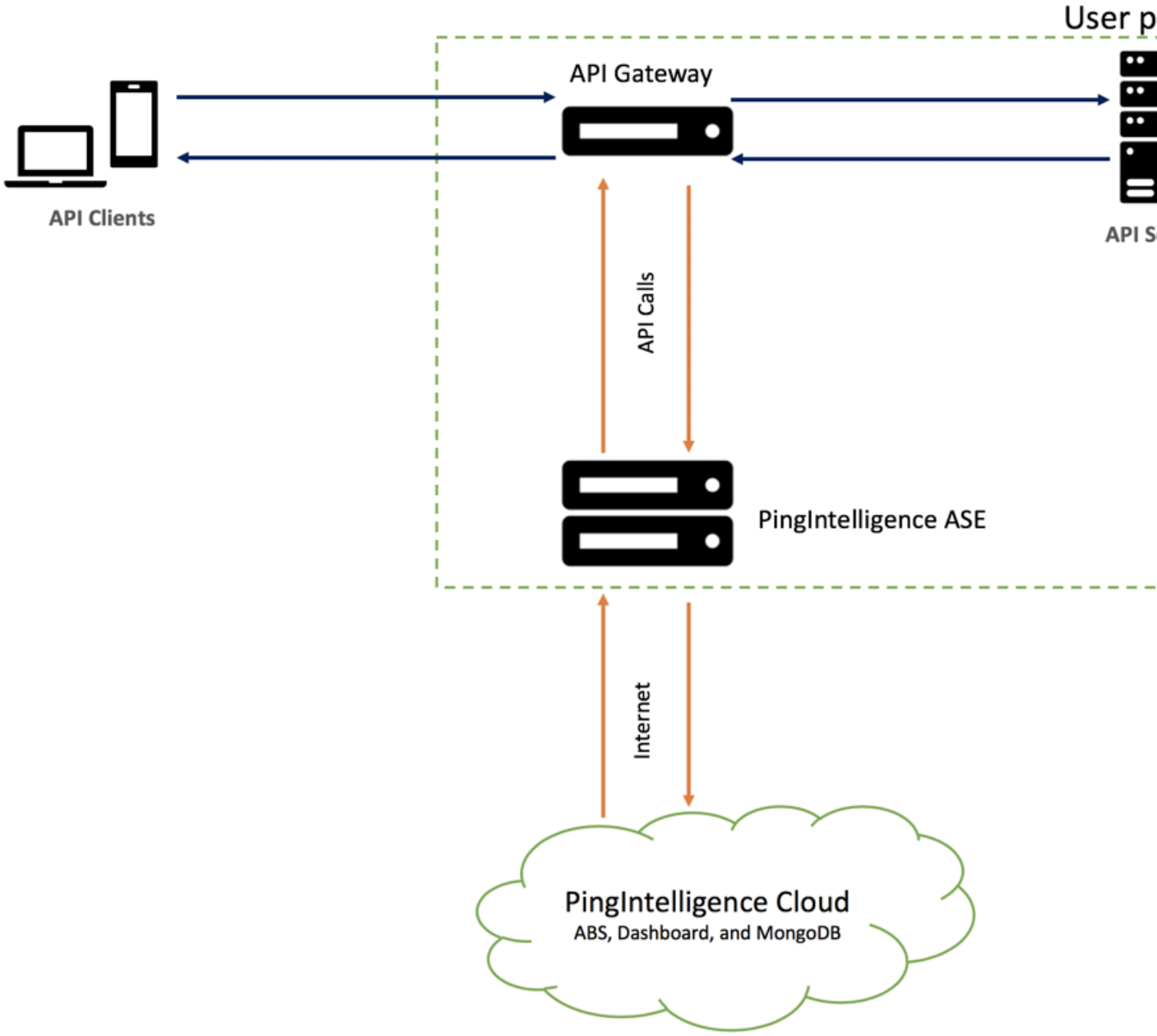
mode, set the `mode=inline` in the `ase.conf` file. The following diagram shows the inline deployment:



Sideband Mode

In sideband mode, ASE receives API calls from an API gateway which uses policies to send API request and response metadata to ASE. In this mode, the API Gateway still terminates the client requests and manages the traffic flow to the API servers. PingIntelligence currently supports sideband policies on the following platforms, *PingIntelligence API gateway integrations.* .

To configure ASE to work in sideband mode, set the `mode=sideband` in the `ase.conf` file. The following diagram shows the sideband mode of deployment:



For more informatio on different ASE modes, see the *ASE Admin Guide*.

Download and install ASE software

About this task

ASE supports RHEL7.6 or Ubuntu 16.04 LTS on an EC2 instance, bare metal x86 server, and VMware ESXi. You can install ASE as a root or a non-root user. You can install ASE either by downloading the ASE software from the download site or by using the ASE Docker image provided to you.

Install ASE by downloading the ASE software

Complete the following steps to install ASE:

Steps

1. Go the *download site*
2. Under PingIntelligence, click on **Select** and navigate to the ASE section to download the ASE binary. Make sure you choose the correct platform binary.
3. After downloading the file, copy the ASE file to the `/opt` directory if you are installing as a root user. You can choose any other location if you want to install ASE as a non-root user.
4. Change the working directory to `/opt`
5. At the command prompt, type the following command to untar the ASE file:

```
tar -zxvf <filename>
```

For example:

```
tar -zxvf ase-rhel-4.0.1.tar.gz
```

6. To verify that ASE successfully installed, the `ls` command at the command prompt. This will list the pingidentity directory and the build's tar file. For example:

```
/opt/pingidentity/ase/bin/$ ls
pingidentity ase-rhel-4.0.1.tar.gz
```

ASE License

To start ASE, you need a trial license which is valid for 30-days. At the end of the trial period, ASE stops accepting the traffic.

ⓘ **Note:** Contact PingIdentity sales to get an ASE trial license.

Configure ASE license

After receiving the ASE license key, download and save the license file as `PingIntelligence.lic`. Copy the license file to the /opt/pingidentity/ase/config directory and start ASE.

Update an existing license If your license expires, obtain an updated license from PingIntelligence for APIs sales and replace the license file in the `/opt/pingidentity/ase/config` directory. Stop and then start ASE to activate the new license.

Configure PingIntelligence Cloud Connection

Navigate to `/opt/pingidentity/ase/config/abs.conf` and refer to the PingIntelligence cloud information received via email to configure the following:

- Set `abs_endpoint` to ABS IP
- Set `access_key` to ABS access key
- Set `secret_key` to ABS secret key
- Set `enable_ssl` to true

Here 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.

; 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=
```

Obfuscate access and secret key

Using the ASE command line interface, obfuscate the access key and secret key in `abs.conf`. The access key and secret key has been sent to you through the PingIdentity welcome email.

ASE ships with a default master key (`ase_master.key`) which is used to obfuscate other keys and passwords. You can generate your own ase_master.key. For more information, see *Obfuscate key and passwords*

> ⓘ **Note:** During the process of obfuscation password, ASE must be stopped.

Obfuscate access and secret keys

Enter the access key and secret key provided to you in clear text in `abs.conf`. Run the `obfuscate_keys` command to obfuscate:

```
/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
```

Start ASE after keys are obfuscated.

> ⓘ **Important:** `ase_master.key` must be present in the `/opt/pingidentity/ase/config/` directory for ASE to start.

Start and stop ASE

Start ASE

PrerequisiteFor 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.

Change working directory to `bin` and run the `start.sh` script.

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 4.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Stop ASE

Change working directory to `bin` and 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
```

Enable ASE to ABS engine communication

To start communication between ASE and the AI engine, run the following command:

```
./cli.sh enable_abs -u admin -p
```

To confirm an ASE Node is communicating with ABS, issue the ASE status command:

```
/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                  : 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
abs_attack_request_minutes=10
```

Integrate PingIntelligence into your API environment

Sideband configuration

If you configured PingIntelligence ASE for sideband connectivity with an API Gateway, then refer to the deployment guide for your environment:

- *Akana API gateway sideband integration* on page 489
- *PingIntelligence Apigee Integration* on page 513
- *PingIntelligence AWS API Gateway Integration* on page 531
- *Axway sideband integration* on page 549
- *Azure APIM sideband integration* on page 573
- *PingIntelligence - CA API gateway sideband integration* on page 582
- *F5 BIG-IP PingIntelligence integration* on page 591
- *IBM DataPower Gateway sideband integration* on page 603
- *PingIntelligence - Kong API gateway integration* on page 610
- *Mulesoft sideband integration* on page 616
- *NGINX sideband integration* on page 631

- *NGINX Plus sideband integration* on page 646
- *PingAccess sideband integration* on page 665
- *PingIntelligence WSO2 integration* on page 677

After completing the setup steps in the integration guide, go to AI Engine training.

Configure ASE and Dashboard

To configure the ASE system and Dashboard to work with PingIntelligence cloud, use the configuration details that you received in an email from PingIntelligence. The following details have been emailed to you:

ABS configuration

- ABS IP
- ABS access key
- ABS secret key

Dashboard Configuration

- Dashboard IP
- Dashboard username
- Dashboard password

Add APIs to ASE

To secure an API with PingIntelligence for APIs software, an administrator can add an API definition to the Ping Identity ASE which will then pass the API information to the AI Engine for reporting and attack detection. Complete the following steps to configure a simple REST API. For more information on advanced options, see the *ASE Admin Guide*.

1. Navigate to `/opt/pingidentity/ase/config/api` and copy the file `rest_api.json.example` to `rest_api.json`
2. Open the `rest_api.json` file and update the following information:
   a. Update the “url” to the base path of the API, for example, `"/apiname"`
   b. Replace the server IP addresses and ports with the addresser/ports of your app servers.
   c. Review the following parameter list and make other edits as applicable.

Key API JSON file parameters to configure include:

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 t `"/shopping"`– name of a 1 level API `"/shopping/electronics/phones"` –3 level API `"/"` – entire server (used for ABS API Discovery or load balancing)
hostname	Hostname for the API. The value cannot be empty. `"*"` matches any hostname.
cookie	Name of cookie used by the backend servers.

oauth2_access_token	When `true`, ASE captures OAuth2 Access Tokens. When `false`, ASE does not look for OAuth2 Tokens. Default value is For more information, see *Configuring OAuth2 Token*.
apikey_qs	When API Key is sent in the query string, ASE uses the specified para For more information, see *Configuring API Keys*.
apikey_header	When API Key is part of the header field, ASE uses the specified para For more information, see *Configuring API Keys*.
login_url	Public URL used by a client to connect to the application.
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 determ
health_retry_count	The number of times ASE queries the backend server status after not
health_url	The URL used by ASE to check backend server status.
server_ssl	When set to true, ASE connects to the backend API server over SSL.
Servers:	The IP address or hostname and port number of each backend serve
host	See *REST API Protection from DoS and DDoS* for information on opt
port	
server_spike_threshold	
server_connection_quota	
The following API Pattern Enforcement parameters only apply when API Firewall is activated	
Flow Control	ASE flow control ensures that backend API servers are protected fron
client_spike_threshold	See *WebSocket API Protection from DoS and DDoS* for information o
server_connection_queueing	
bytes_in_threshold	
bytes_out_threshold	
protocol_allowed	List of accepted protocols Values can be HTTP, HTTPS, WS, WSS. ⓘ **Note:** When Firewall is enabled, protocol_allowed takes precede
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 exa

Decoy Config	When decoy_enabled is set to `true`, decoy sub-paths function as de
decoy_enabled	response_code is the status code (for example, 200) that ASE returns
response_code	response_def is the response definition (for example OK) that ASE re
response_def response_message	response_message is the response message (for example OK) that A
decoy_subpaths	decoy_subpaths is the list of decoy API sub-paths (for example `shop`
	See *API deception* for details

After configuring the API JSON file, add it to ASE for it to take effect. To add a runtime API, execute the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh add_api {file_path/api_name} -u admin -p
```

Verify/List the API

To verify whether the API that you added has been successfully added or not, run the list API command:

```
opt/pingidentity/ase/bin/cli.sh list_api -u admin -p
```

AI engine training

The PingIntelligence AI Engine needs to be trained before it can detect anomalies or attacks on API services or generate reports. The AI training runs until a minimum amount of data is received, and the training period is completed for the given API.

ABS must be trained on all APIs before they can be secured. Whenever a new API is added, ABS automatically trains itself before looking for attacks

For detailed information on training the AI Engine, see the *ABS Admin guide*.

Connect to the PingIntelligence dashboard

The PingIntelligence Dashboard provides information on the APIs monitored by PingIntelligence for APIs. Until the training period is complete (based on volume of traffic) for an API, only a minimal amount of Dashboard data will be available. If traffic volume is low, it may take several days before many of the Dashboard graphs have data.

To connect to the Dashboard and work with PingIntelligence cloud, use the connection details that you received in the welcome email from Ping Identity. The following details are emailed to you :

- Dashboard URL - It is used to load the PingIntelligence for APIs Dashboard
- Dashboard User Name
- Dashboard User Password

For more information on accessing and using Dashboard, see *Access the PingIntelligence Dashboard* .

For more information on PingIntelligence for APIs Dashboard, see *PingIntelligence Dashboard*.

Access 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 templates to use Postman, a free tool for formatting REST API reports.

ⓘ **Note:**

Until the training period is complete (based on volume of traffic) for an API, only a minimal amount of reporting data will be available. If traffic volume is low, it may take several days before some of the reports (e.g. attack reports) have data.

Install Postman with PingIntelligence for API reports

Ping Identity provides configuration files which 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 certificate with Postman

ABS ships with a self-signed certificate. To use Postman with the self-signed certificate of ABS, disable the certificate verification option by following the steps at *this link*

View ABS reports in Postman

To view the reports in Postman, complete the steps mentioned in the *View ABS reports in Postman* topic. In configuring the environment, the following details are required:

1. `Server`: Use the ABS URL provided in the email
2. `Port`: Use the port number located at the end of the ABS URL in the email
3. `Access_Key`: Use the ABS access key provided in the email
4. `Secret_key`: Use the ABS secret key provided in the email

`API_Name`is the name of the API. Do not edit any variables that start with "`system`".

ⓘ **Note:** For detailed information on ABS reports, see *Attack Reporting* in the ABS Admin Guide.

Following is a list of reports that you can generate using Postman or any other REST API client:

- Metrics report
- Anomalies report
- API key metrics report
- OAuth2 token metrics report
- OAuth2 token forensics report
- IP forensics report
- Cookie forensics report
- Various attack types
- Flow control report
- Blocked connections report
- Backend error report
- List of valid URLs
- List of hacker's URLs

PingIntelligence Production Deployment

Automated deployment

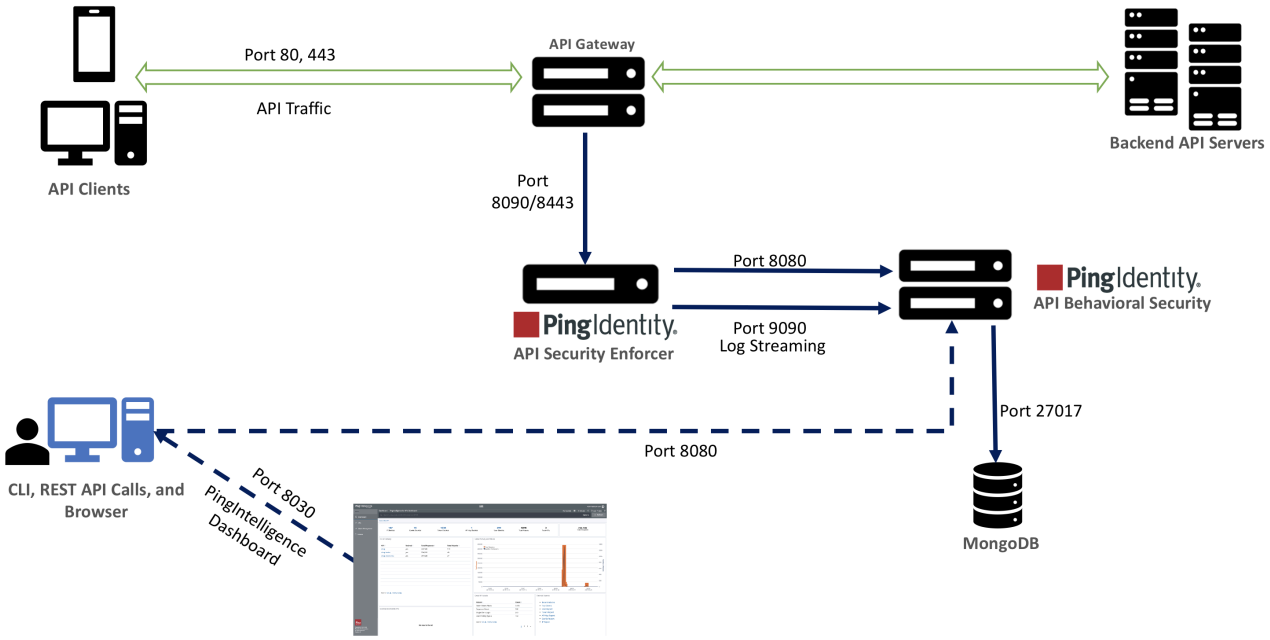
PingIntelligence for APIs setup

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 two platforms: API Security Enforcer and API Behavioral Security Artificial Intelligence engine.

This guide describes the installation and execution of an Ansible package which automatically builds a PingIntelligence for APIs environment with PingIntelligence for RHEL 7.6 or Ubuntu 16.04 LTS. The package installs and configures the following components:

- ASE (deployed between the API clients and API Gateway or backend server)
- ABS AI Engine
- MongoDB database
- PingIntelligence for APIs Dashboard

The following diagram shows the complete deployment architecture of the



Default ports used by automated deployment. These ports can be configured using default `yml` files.

setup.

API Security Enforcer (ASE)

Applies real-time inline inspection of API traffic to detect and block attacks. ASE works with the ABS engine to identify attacks.

API Behavioral Security (ABS)

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

PingIntelligence for APIs Dashboard offers you the following:

- View the various APIs in your API environment along with the API creation date
- View the training status and other information of your APIs
- View your API dashboard
- Unblock a specific client identifier
- Tune threshold
- 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 blocked connections. The dashboard engine makes periodic REST API calls to an ABS AI engine which returns JSON reports that are used to generate graphs in PingIntelligence Dashboard.

Users

You can install all the PingIntelligence products 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 products should be installed as a `sudo` user or as a normal user.

Time zone

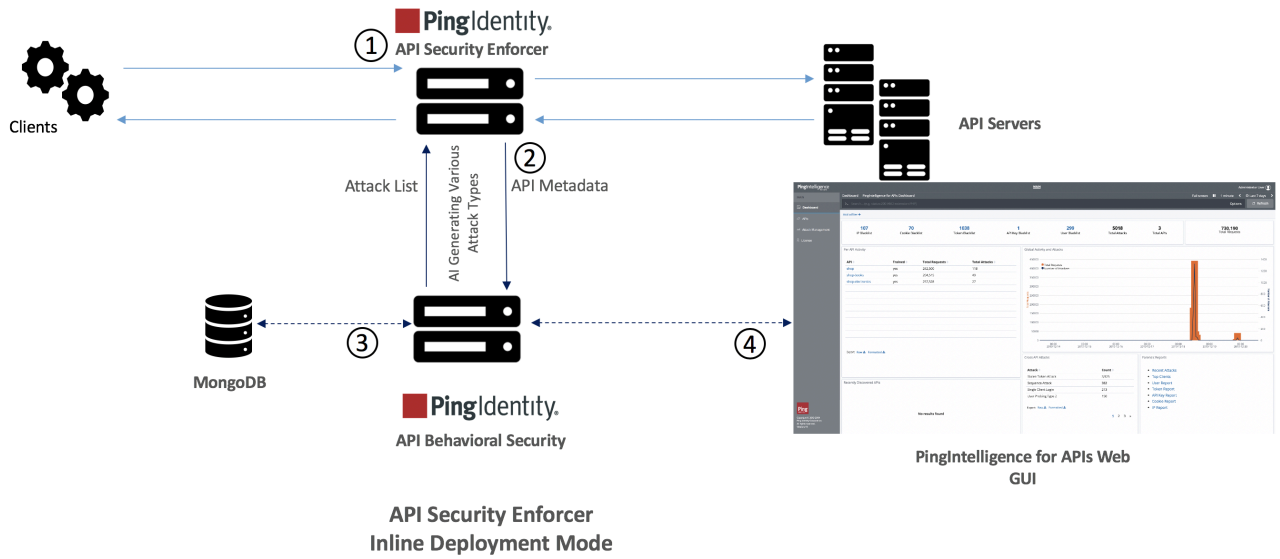
All the PingIntelligence products namely ASE, ABS, and PingIntelligence Dashboard should be in the same timezone, either local or UTC. Make sure that the third-party product, MongoDB, is also in the same timezone as PingIntelligence products.

PingIntelligence deployment modes

Inline mode

In PingIntelligence **inline** deployment mode, API Security Enforcer (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 the inline mode, ASE deployed at the edge of the datacenter, terminates SSL connections from API clients. It then forwards the requests directly to the correct APIs – and app servers such as Node.js, WebLogic, Tomcat, PHP, etc.

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



API Security Enforcer Inline Deployment Mode

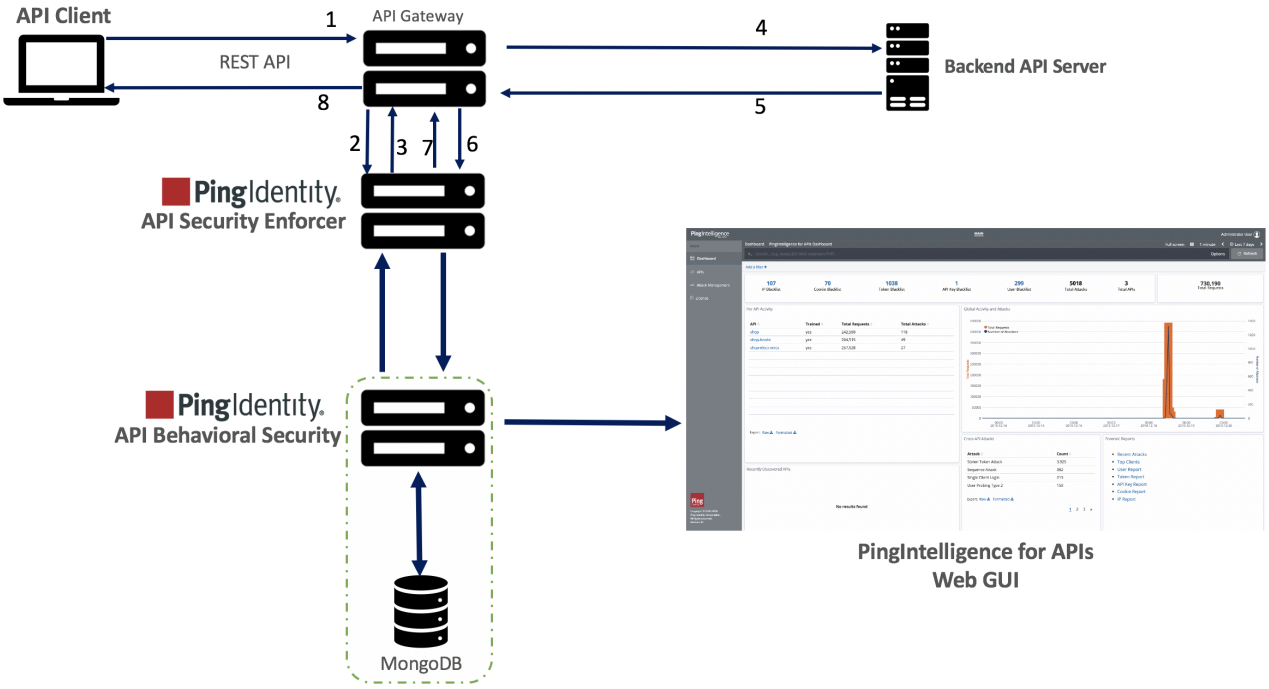
Following is a high-level description of traffic flow:

1. Client request is received by ASE. The request is logged in 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 is sent to ABS AI engine for processing. 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. PingIntelligence for APIs Web GUI fetches the data from ABS to display in the dashboard.

Sideband mode

When PingIntelligence is deployed in the **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 the sideband mode, set the **mode=sideband** in the `ase-defaults.yml` file.



API Security Enforcer
Sideband Deployment Mode

Following is a description of the traffic flow through the API gateway and Ping Identity ASE.

1. Incoming request to API gateway
2. 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 Blacklist. If all checks pass, ASE returns a 200-OK response to the API gateway. Else, 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, else 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:** Complete the ASE sideband mode deployment by referring to API gateway specific deployment section on the *PingIntelligence documentation site*.

Prerequisites

Commonly used terms for deployment machines

Following terms are frequently used during automated deployment steps:

- **Management host** - Management host machine or a management machine is the machine where the PingIntelligence automated deployment is downloaded and run.
- **Host machine** - A host machine is the machine where PingIntelligence components are installed.

Prerequisite

The following prerequisites must be met before proceeding with the installation:

- **Management machine operating system** - Automated deployment requires RHEL 7.6 operating system on the management host.
- **Host machine operating system** - Host machine operating system can be RHEL 7.6 or Ubuntu 16.04 LTS.

> ⓘ **Important:** Make sure that the deployment is homogenous with respect to the provisioned host machines. Either all the host machines should be running RHEL 7.6 or Ubuntu 16.04 LTS. Do not create a set up having both types of host machines.

- **Ansible** - The management host machine should have ansible 2.6.2 installed
- **Python** - The management host machine should have Python 2.7
- **User** - Automated installation requires a user with password-less authentication for SSH connection to the host machines. User should also have password-less `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 `hosts` file, see *Step 3 - Configure hosts file and download software* on page 50.
- **fiewalld package** - All the host machines should have an active `firewalld [python 2.7]` 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 the respective *Change default settings* topics.
- If you are deploying the setup on a Ubuntu machine, make sure that the MongoDB host machine has `libcurl4-openssl-dev`.

Download the deployment package

Setup the management host

PingIntelligence automated deployment requires RHEL 7.6 management host machine to start the deployment. The automated deployment installs different PingIntelligence components from this management machine.

1. Login 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-4.2.tar.gz
```

Untarring the file creates the following sub-directories in the `pi-api-deployment` directory:

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 ASE, 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, then 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 password-less.

`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 any of its content.
`util`	Contains utilities to run PingIntelligence components as a service.
`external`	The third-party components like MongoDB are downloaded in the `external` directory.
`keys`	After the installation is complete, the master keys of all the products are saved here.
`license`	Contains `ase` and `abs` directories that have the ASE and ABS 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 `updated_packages` sub-directory which stores the PingIntelligence updated binaries with new master keys. You can use these binaries for future use.

Step 1 - User and authentication

This covers concept and steps to create an SSH user and configure password-less authentication for the SSH user or using a password to connect to the host machines. Creating a new user is an optional step. You can use the default user configured in the `hosts` file.

- *User creation (optional)*
- *Authentication*

User Creation (Optional)

Complete the following steps on all the provisioned host machines if you do not have a user as mentioned in the prerequisites section. If you already have a user as described in the prerequisite section, you can skip the following steps:

1. Create ec2-user. The `hosts` file in the automation package has `ec2-user` as the default user. You can create your own username.

   ```
   #useradd ec2-user
   ```

2. Change the password

   ```
   #passwd ec2-user
   ```

3. ⓘ **Note:** If you plan to install PingIntelligence software as a `non-sudo` user, then skip steps 3-5.

   Add the user to the wheel group

   ```
   #usermod -aG wheel ec2-user
   ```

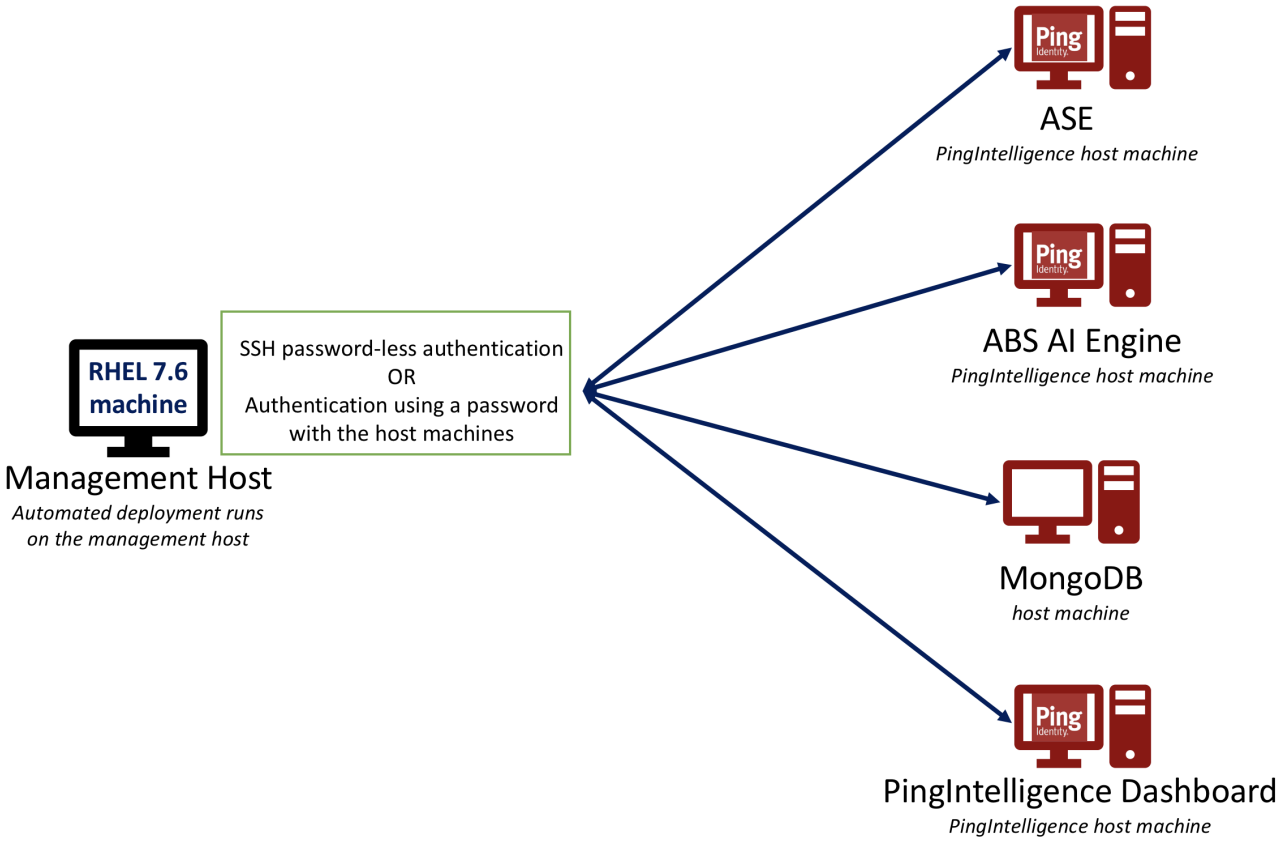
4. Configure password-less `sudo` access

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

5. Verify the `/etc/ssh/sshd_config` file for `PubKeyAuthentication`. If it is set to no, then set it to yes and restart `sshd` service using the following command:

   ```
   #systemctl restart sshd
   ```

The following diagram shows the management host and PingIntelligence host machines communicating either through password-less SSH communication or communicating after authenticating using a password.



Authentication

PingIntelligence automated deployment provides the following two methods for authentication between the management host machine and PingIntelligence host machines.

- ***Password-less authentication*** - There are two methods to achieve password-less authentication.
- ***Authentication using a password*** - Authentication using a password requires `sshpass` module to be installed on the RHEL host machine.

Password-less authentication

You can set up a password-less authentication from the management machine to other machines where PingIntelligence components are installed. There are two possible methods to configure password-less authentication.

Method 1

1. Run the following command on the management machine. The management machine is the machine from which the automated deployment script is run to deploy the various PingIntelligence software.

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

   This command generates the `ssh-keys`. Accept all the default options. Make sure that you do not set the password for the key.
2. Run the following command for each VM except the Ping Management VM:

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

   For example, `ssh-copy-id pi-user@192.168.11.148` (ping-ase)

Method 2

1. Run the following command on the management machine. The management machine is the machine from which the automated deployment script is run to deploy the various PingIntelligence software.

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

   This command generates the `ssh-keys`. Accept all the default options. Make sure that you do not set the password for the key.
2. Fetch the generated key in step 1 from `/home/$USER/.ssh/id_rsa.pub`
3. Copy and add this key in the `/home/$USER/.ssh/authrorized_keys` file on all the machines where PingIntelligence components are installed.

ⓘ **Important:** If method 1 or method 2 of configuring password-less authentication does not succeed, contact your system administrator.

Authentication using a password

You can also use password to authenticate with PingIntelligence and MongoDB host machines. Configure the password of the host machine in the `hosts` file. Complete the following prerequisites to authenticate using a password:

Prerequisites:

- Install `sshpass` module on the management host machine. Note that 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.

To add the password in the `hosts` file, edit the `hosts` file to configure password in ansible_ssh_pass parameter as shown in the `hosts` file snippet below.

```
# 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>
```

Verify SSH connectivity

You can manually verify SSH connectivity between the management host machine and the PingIntelligence machine by entering the following command.

```
ssh user@remote-machine "ls"
```

Step 2 - Configure licenses

PingIntelligence ASE and ABS require a valid license to start. The license file for both the products is named `PingIntelligence.lic`.

- **ASE:**

  Copy the ASE license file in the `license/ase` directory. Make sure that the license file is named as `PingIntelligence.lic` Following is a sample of the ASE license file:

  ```
  ID=981894
  Product=PingIntelligence
  Module=ASE
  Version=4.2
  IssueDate=2020-07-01
  EnforcementType=0
  ExpirationDate=2020-12-30
  Tier=Subscription
  SignCode=
  Signature=
  ```

  Verify that the correct file has been copied: To verify that the correct license file has been copied in the `/license/ase` directory, run the following command:

  ```
  # grep 'Module' license/ase/PingIntelligence.lic
  Module=ASE
  ```

- **ABS:**

  Copy the ABS license file in the `license/abs` directory. Make sure that the license file is named as `PingIntelligence.lic`. Following is a sample of the ABS license file:

  ```
  ID=981888
  Product=PingIntelligence
  Module=ABS
  Version=4.2
  IssueDate=2020-07-01
  EnforcementType=0
  ExpirationDate=2020-12-30
  Tier=Subscription
  SignCode=
  Signature=
  ```

  Verify that the correct file has been copied: To verify that the correct license file has been copied in the `/license/abs` directory, run the following command:

  ```
  # grep 'Module' license/abs/PingIntelligence.lic
  Module=ABS
  ```

Step 3 - Configure hosts file and download software

The hosts file contains the various parameters to be configured for installation of PingIntelligence components. Complete the following steps to configure the hosts file.

The configuration file has parameters where link to download third-party component is configured. 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. All the PingIntelligence components should either be installed on an RHEL machine or on Ubuntu machines.

Configure the following fields in the `config/hosts` file:

Variable	Description
IP addresses ▪ `[ase]` ▪ `[abs]` ▪ `[mongodb]` ▪ `[dashboard]` ▪ `[elasticsearch]` ▪ `[kibana]` ▪ `[abs_reporting_node]` ▪ `[webgui]`	Configure the following 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 ▪ `[kibana]` - Kibana 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]` If you are setting up a POC environment, then all the components: ASE, ABS, MongoDB, Dashboard, WebGUI, ElasticSearch, and Kibana can share a single IP address. ⓘ **Note:** 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, Kibana, and ABS Reporting node (optional) can be deployed on a single server. ▪ ElasticSearch should be deployed on a standalone server.

`installation_path`	Configure the path where you would want the PingIntelligence components to be installed. 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, then 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 it to `true` if you want automated deployment to install MongoDB. Set it to `false` if you want to use an existing MongoDB installation. 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 `abs.properties` file.
`install_elasticsearch`	Set to `true` if you want automated deployment to install Elasticsearch. Set it to `false` if you want to use an existing Elasticsearch installation. Default value is `true`. Note the following points: ▪ If you have 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 URL of your existing Elasticsearch in `dashboard-defaults.yml` file. For more information, see *Change Dashboard default settings* on page 61. ⓘ **Note:** If you are using an existing Elasticsearch installation is an OSS package, make sure that Kibana 6.8.1 OSS package is available in `external` directory.
`jdk11_download_url`	The automated script requires `OpenJDK 11.0.2`. ⓘ **Note:** If your machine does not have internet access, then download the OpenJDK 11.0.2 and save the file as `openjdk11.tar.gz` in `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, then download the MongoDB 4.2.0 and save the file as `mongodb.tgz` in `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, then download the Elasticsearch 6.8.1 and save the file as `elasticsearch-6.8.1.tar.gz` in `external`directory.
`kibana_download_url`	Kibana download URL. A default URL is populated in the `hosts` file. ⓘ **Note:** If your machine does not have internet access, then download the Kibana 6.8.1 and save the file as `kibana-6.8.1.tar.gz` in `external`directory.
`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 password-less authenticated connection.

Add Ansible username in the `ansible_ssh_user` field. The default value is `ec2-user`.

```
[ase]
172.16.40.81

[abs]
172.16.40.81

[abs_reporting_node]


[mongodb]
172.16.40.81 mongo_port=27017

[dashboard]
172.16.40.81

[elasticsearch]
172.16.40.81
```

```
[kibana]
172.16.40.81

[webgui]
172.16.40.81

[all:vars]

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

# install_as_service set to true will start ase, abs, aad, dashboard,
 elasticsearch
# and kibana 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 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

# 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-
rhel70-4.2.0.tgz'
elasticsearch_download_url='https://artifacts.elastic.co/downloads/
elasticsearch/elasticsearch-6.8.1.tar.gz'
kibana_download_url='https://artifacts.elastic.co/downloads/kibana/
kibana-6.8.1-linux-x86_64.tar.gz'

# 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=
```

Manually download third-party components

The automated deployment downloads the third-party packages when it is executed. However, if your Management host machine does not have internet access, then download the software using the steps mentioned below. Download the individual components and save the file in the `external` directory.

ⓘ **Important:** If your management host machine has internet access, you can skip downloading the third-party components manually.

1. Install Ansible version 2.6.2 on the Management host machine. The Management host is the machine from where the automated deployment script is run to deploy the various PingIntelligence software.
2. Install Python 2.7 on the Management host machine.
3. Download the following packages and copy to the `external` directory using the specified names:

   MongoDB – Download MongoDB 4.2 from:

   - **Linux:** *https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.2.0.tgz* and save the file in the `external` directory as `mongodb.tgz`.
   - **Ubuntu:** *http://downloads.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1604-4.2.0.tgz* and save the file in the `external` directory as `mongodb.tgz`.

   Elasticsearch – Download Elasticsearch from: *https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.8.1.tar.gz* and save the file in the `external` directory as `elasticsearch-6.8.1.tar.gz`.

   Kibana – Download from: *https://artifacts.elastic.co/downloads/kibana/kibana-6.8.1-linux-x86_64.tar.gz* and save the file in the `external` directory as `kibana-6.8.1-linux-x86_64.tar.gz`.

Download PingIntelligence for APIs software

Download the following PingIntelligence for APIs software to `pi-api-deployment/software` directory.

- API Security Enforcer (RHEL 7.6 or Ubuntu 16.0.4 LTS)
- API Behavioral Security
- 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-4.2.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 159M Jun 07 00:01 pi-api-
abs-4.2.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 38M  Jun 07 00:01 pi-api-ase-
rhel-4.2.tar.gz
```

Check SSH connectivity

About this task

Check the SSH connectivity from the management host machine to other host machines. The SSH connectivity check provides details regarding the configured `user`, IP address of the hosts for which SSH connectivity works or fails. Run the check before deploying PingIntelligence components. Enter the following command on the management host command line.

Steps

$ ./bin/start.sh check

```
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
```

Few possible errors during SSH connectivity

During SSH connectivity check between management host machine and PingIntelligence hosts, you may encounter some errors because of user permission issues or connectivity issues between machines. Following are some of the probable error messages that you may see:

- You have configured user to use 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": "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
```

- `ansible_ssh_pass` for authentication with password is uncommented in the `hosts` file, however, the field has been left empty. Leaving the value empty is equivalent to passworld-less authentication.

```
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
```

- `install_with_sudo` is set to `true` and there is an error connecting to PingIntelligence host machines.

```
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
```

  The probable reasons for error in connectivity could be:

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

```
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
```

- **selinux** dependency - If you encounter the following error, you need to install `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.

```
[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
```

Change default settings

The deployment package provides `yml` files to change the default settings of ASE, ABS, Web GUI, and AAD. It is recommended to change the default settings before you execute the deployment package. For more information on each component, see the respective guides at *PingIntelligence documentation* site. The following topics describe the default settings of each component:

- *Change ASE's default settings*
- *Change ABS default settings*
- *Change Dashboard default settings* on page 61
- *Change AAD default settings*

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

Change ASE's 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 the sideband mode.
`http_ws_port`	Data port used for HTTP or WebSocket protocol. The default value is 8090.
`https_wss_port`	Data port used for HTTPS or secure WebSocket protocol. The default value is 8443.
`management_port`	Management port used for CLI and REST 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`.
`enable_sideband_keepalive`	This key is used only in ASE sideband mode. Setting it 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 at 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 to know the current password.
`timezone`	Defines ASE's timezone. The possible values are `local` or `utc`

ⓘ **Important:** Make sure to take a backup of the `ase-defaults.yml` file on a secure machine after the automated installation is complete.

Following is a sample `ase-defaults.yml` file:

```
---
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: 8090
    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

    # enable keepalive for ASE in sideband mode
    enable_sideband_keepalive: false
```

```
# 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

# timezone setting
# allowed values: local, utc
timezone: local
```

Change ABS default settings

You can change the default settings in ABS by editing the `abs-defaults.yml` file. 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.
`log_port`	Port for ASE to send log files to ABS. The default value is 9090.
`mongo_username` and `mongo_password`	MongoDB user name and password. The default user name 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`	Set it to `true` if MongoDB is configured to use SSL connections. The default value is `false`. PingIntelligence deployment ships with a default self-signed certificate.
`mongo_replica_set`	Name of the MongoDB replica set. Default name is `absrs01`.
`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 JKS Keystore. 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 CLI admin is `admin`. To change the password, you need to know the current password.
`poc_mode`	Sets the mode in which 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 PoC deployment.

ⓘ **Important:** Make sure to take a backup of the `abs-defaults.yml` file on a secure machine after the automated installation is complete.

Following is a sample `abs-defaults.yml` file:

```
---
abs:
  # Define ports for the PingIntelligence ABS
  # Make sure ports are not same for single server installation
  management_port: 8080
  log_port: 9090

  # 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: false
  # Mongo replica set name
  mongo_replica_set: absrs01

  # 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

# 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
```

Change the default system memory Complete the following steps to change the default system memory in `abs.properties` file of ABS.

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

```
# tar –zxvf pi-api-abs-4.1.tar.gz
```

3. Edit the `abs.properties` file in `config` directory to change the default value of `system_memory` to 50% of host memory. For example, if host ABS system has 16 GB of memory, set the value to 8192 MB.

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

4. Save the file
5. Tar the ABS binary and save it with the same file name (`pi-api-abs-4.1.tar.gz`) in `software` directory by entering the following command:

```
# tar -czf pi-api-abs-4.1.tar.gz pingidentity/abs
```

Change Dashboard default settings

You can change the default settings of PingIntelligence Dashboard by editing the `dashboard-defaults.yml` file. The following table lists the variables that you can set for PingIntelligence Dashboard:

Variable	Description
`port`	Port number to connect to PingIntelligence Dashboard.
`admin_password` and `ping_user_password`	The password for `admin` and `ping_user`.
SSL configuration for PingIntelligence Dashboard ▪ `server_ssl_key_store_password` ▪ `server_ssl_key_alias`	Configure the passwords for keystore and key alias.

H2 database configuration: ▪ `h2_db_password` ▪ `h2_db_encryption_password`	Password for H2 database and password for encryption
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, see*Discovered APIs* on page 471 ▪ `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 X-pack. 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 URL. Enter the complete URL including http/https. For example, https://myelasticsearchurl.pi.com:443. Providing the port number in the URL is mandatory.
`elastic_username`	If you want to use an already available Elasticsearch username, configure it in `elastic_username`.
`kibana_port`	The port number on which Dashboard communicates with Kibana..
`elastic_password`	Elasticsearch password. The default value is `changeme`. ⓘ **Note:** Do not change the `elastic_password` after PingIntelligence installation is complete.

`kibana_password`	Kibana password. The default value is `changeme`. ⓘ **Note:** Do not change the `kibana_password` after PingIntelligence installation is complete.
`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 syslog server.
`restricted_user_access`	Defines the user for viewing information in API Dashboard. Set it to `true` to set the user as a restricted user. The header in API query string used depends on the type of user, restricted or admin. For more information on user headers, see *ABS users for API reports* on page 286

ⓘ **Important:** Make sure to take a backup of the `dashboard-defaults.yml` file on a secure machine after the automated installation is complete.

Following is a sample `dashboard-defaults.yml` file:

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

  # webgui "admin" account password
  admin_password: changeme
  # webgui "ping_user" account password
  ping_user_password: changeme

  # 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:

dashboard:
  ui:
    # Install elasticsearch with xpack enabled
    # If there is no authentication on pre-existing elasticsearch, set this
 to false
    enable_xpack: true

    # 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, dashboard, webgui and kibana nodes.
    elasticsearch_url: https://search-giuijfysjfsxucty.pingidentity.com:443

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

    # Passwords for "elasticsearch", "kibana", "ping_user" and "ping_admin"
 users
    # Dashboard will be accessible for these accounts
    # Please set strong passwords
    # If enable_xpack is set to false, below passwords are ignored
    elastic_password: changeme
    kibana_password: changeme
    ping_user_password: changeme
    ping_admin_password: changeme

    # Define ports for the PingIntelligence Dashboard
    # Make sure ports are not same for single server installation
    kibana_port: 5601

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

  # ABS Restricted user access ( true/false )
  # Set to false for displaying non-obfuscated blacklist in Kibana
  abs:
    restricted_user_access: false
```

Step 4 - Configure system parameters

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

- `vm.max_map_count`: For Elasticsearch
- `ulimit`: For ASE and Elasticsearch

Run the following command to configure the system parameters on the respective VMs. The script uses `sudo` access for the user on the Elasticsearch and ASE hosts. The IP address of these hosts was configured in the `hosts` file in *Step 1*. 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.
```

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
================================================================================
```

Manually configuring the system parameters

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

1. Set the `vm.max_map_count` to 262144 on the Elasticsearch VM. To set the count, enter the following command:

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

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

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

2. Set the `ulimit` to 65536 on the ASE and Elasticsearch VMs. To set the `ulimit`, complete the following:

   edit `/etc/security/limits.conf` for increasing the soft limit and hard limit. 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
   ```

Step 5 - Install the PingIntelligence for APIs software

Run the following command to setup the deployment. Accept the EULA displayed on the screen for ABS for installation to start.

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

To verify a successful setup, view the `ansible.log` file. Here 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
================================================================================
```

Updated PingIntelligence packages

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. 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.

ⓘ **Note:** The 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.

Install PingIntelligence as a systemd service

You can install the various PingIntelligence components as a systemd service. Installing as a service, the various components are started automatically when the host system restarts. You require `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. Run the following command on the host machine for which you want to verify that is service is installed or not:

```
# 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.
```

Prerequisite for installing PingIntelligence service:

- 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`
  - **Kibana**: `pi-kibana.service`
- Stop the component for which you want to install the service.

Steps: Complete the following steps:

1. Make sure that the component for which you want to install the service is stopped.
2. Log in to the host machine for which you want to install the service. For example, if you want to install ASE as a service, log in to the ASE host machine.
3. 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>
   ```

   For example, on ASE host machine:

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

   Install service for each component in a similar way on the respective host machine.

Order of restarting PingIntelligence components: Edit the service files to make sure that PingIntelligence components in the following order. Use the `Requires` option to set the order of starting of service. For more information, see *Creating and modifying systemd unit files* :

1. MongoDB
2. ABS
3. ASE
4. Elasticsearch
5. Kibana
6. Dashboard
7. Web GUI

Verify PingIntelligence Installation

Verify that all the components have installed and started successfully.

Verify ASE installation

Log in to the ASE host machine and navigate to `<installation-path>/pingidentity/ase/bin` directory and run the `status` command:

```
/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
```

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

Verify ABS and MongoDB installation

Log in to the ABS EC2 instance and run the ABS Admin REST API using a REST API client like Postman. More information on installing and configuring Postman is available in the ABS Admin Guide.

The report can be accessed by calling the ABS system at the following URL:

`https://<abs_ip>:<abs_port>/v4/abs/admin`. Use the IP address from the hosts file.

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


```
{
    "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": ""
            }
        }
    ],
    "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",
                "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
            }
```

```
            ]
        }
}
```

Verify Dashboard Installation

To verify the Dashboard installation, enter the Dashboard IP address from the hosts file in your web browser. Log in using `ping_user` or `admin` username and the password configured in the `dashboard-defaults.yml` file.

See the ASE, ABS and Dashboard guides for configuration and administration of PingIntelligence products.

Next steps - Integrate PingIntelligence into your environment

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

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* on page 489
- *PingIntelligence Apigee Integration* on page 513
- *PingIntelligence AWS API Gateway Integration* on page 531
- *Azure APIM sideband integration* on page 573
- *Axway sideband integration* on page 549
- *PingIntelligence - CA API gateway sideband integration* on page 582
- *F5 BIG-IP PingIntelligence integration* on page 591
- *IBM DataPower Gateway sideband integration* on page 603
- *PingIntelligence - Kong API gateway integration* on page 610
- *Mulesoft sideband integration* on page 616
- *NGINX sideband integration* on page 631
- *NGINX Plus sideband integration* on page 646
- *PingAccess sideband integration* on page 665
- *PingIntelligence WSO2 integration* on page 677

Inline configuration: If you configured PingIntelligence ASE as *Inline ASE* on page 185, the next step is to add *API definitions to the PingIntelligence for APIs* software. After this is complete, direct your API client to the IP address of the ASE software on port 80 or 443.

It is recommended to read the following topics (part of the admin guides) apart from reading the *ASE* and *ABS* Admin Guides:

- *ASE port information*
- *API naming guidelines*
- *Connect ASE and ABS*

After you have added your APIs in ASE, the API model needs to be trained. The training of API model is completed in ABS. The following topics give a high level view, however it is a good practice to read the entire ABS Admin Guide.

- *Train your API model*
- *Generate and view the REST API reports using Postman*: To access the ABS REST API reports you would require the following information:
  - IP address: IP address of ABS configured in the *`config/hosts`* file.
  - Port number: default value is 8080. It is configured in `abs-defaults.yml` file
  - API Name: Name of the API for which you want to generate REST API reports
  - Later and Earlier date: The date range for which you want to generate the reports

- View *Access PingIntelligence Dashboard* on page 17:

  Login to PingIntelligence Dashboard using the `ping_user` login ID and the password that you configured during PingIntelligence installation. For more information on password configuration, see *Change Dashboard default settings* on page 61. The PingIntelligence for APIs Dashboard takes approximately one hour to start showing attack information.

Shut down the deployment

To shut down the deployment and remove all VMs and data, run the `stop.sh` command. When you shut down the deployment, all the VMs along with the data is deleted.

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

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
================================================================================
```

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

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. Following is a snippet of the log file.

```
**************************************** 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
/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
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
 ****************************************
```

Manual deployment

PingIntelligence manual deployment

The topic gives a summary about PingIntelligence products, the different users that can install the product and the time zone in which the products can be deployed.

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 two platforms: API Security Enforcer and API Behavioral Security Artificial Intelligence engine.

API Security Enforcer (ASE)

Applies real-time inline inspection of API traffic to detect and block attacks. ASE works with the ABS engine to identify attacks.

API Behavioral Security (ABS)

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

PingIntelligence for APIs Dashboard offers you the following:

- View the various APIs in your API environment along with the API creation date
- View the training status and other information of your APIs
- View your API dashboard
- Unblock a specific client identifier
- Tune threshold
- 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 blocked connections. The dashboard engine makes periodic REST API calls to an ABS AI engine which returns JSON reports that are used to generate graphs in the PingIntelligence Dashboard.

Users

You can install all the PingIntelligence products 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 products should be installed as a `sudo` user or as a normal user.

Time zone

All the PingIntelligence products namely ASE, ABS, Dashboard, and AAD should be in the same timezone. Make sure that the third-party product, MongoDB, is also in the same timezone as PingIntelligence products.

Part A – Install ABS and MongoDB

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

Install ABS AI engine software

You can install ABS 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.

1. Go to the *download site*
2. Click on **Select** under PingIntelligence
3. Choose the build and click **Download**.

Copy the build file to the `/opt` directory if you are installing the product as a root user. Choose any other location if you want to install ABS as a non-root user.

Install ABS

Before installing ABS, install OpenJDK 11.0.2 on a 64-bit architecture machine with Ubuntu 16.04 LTS or RHEL 7.6. To verify the Java version, run the following command:

```
# java -version
```

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

To install ABS, complete the following steps.

1. Change working directory to `/opt` if you are installing the product as a root user. Choose any other location if you want to install ABS as a non-root user.
2. At the command prompt, type: `# tar -zxvf <file_name>`

   For example, `# tar -zxvf pi-api-abs-4.2.tar.gz`

ABS License

To start 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.
- **Subscription license** – The subscription license is based on the peak number of transactions subscribed for per month and the duration of the license. It is a good practice to *configure your email* before configuring the ABS license. ABS sends an email notification to the configured email ID when the license has expired. Contact the PingIntelligence for APIs sales team for more information. The following points should be noted:
  - **Maximum transaction set to 0**: If your subscription ABS license has zero as maximum transaction, it means that the license has unlimited monthly transaction. Such a license only expires at the end of subscription period.
  - **License expiry**: In case when the subscription license has expired, ABS continues to run until a restart. ABS needs a valid license file to start.

Add an ABS license

If you have not received an ABS license, request a license file from Ping sales. The name of the license file must be `PingIntelligence.lic`. Copy the license file to the`/opt/pingidentity/abs/config` directory and then start ABS.

Update an existing license

If your existing license has expired, obtain a new license from Ping sales and replace the license file in the `/opt/pingidentity/abs/config` directory. Stop and then start ABS after the license file is updated.

Checking the current transaction count

Use the *Admin REST API* on page 309 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
    }
```

Obfuscate passwords

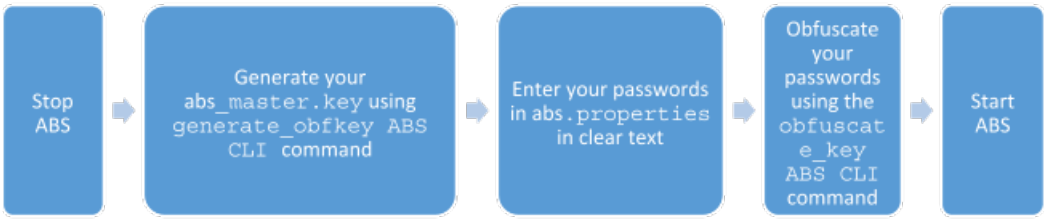
Using ABS command line interface, you can obfuscate the keys and passwords configured in `abs.properties`. The following keys and passwords are obfuscated:

- `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:



Generate abs_master.key

You can generate the `abs_master.key` by running the **`generate_obfkey`** command in the ABS 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
```

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

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

Obfuscate key and passwords

Enter the keys and passwords in clear text in `abs.properties` file. Run the **`obfuscate_keys`** command to obfuscate keys and passwords:

```
/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
```

Start ABS after passwords are obfuscated.

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

Configure SSL

ABS supports only TLS 1.1 and TLS 1.2 and requires Open JDK 11.0.2. You can configure SSL by setting the value of enable_ssl parameter to true in `pingidentity/abs/mongo/abs_init.js` file. Setting the value to true enables SSL communication between ASE and ABS as well as for ABS external REST APIs. Following is a snippet of the `abs.init` file with enable_ssl parameter set to `true`:

```
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"
});
```

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 *Obfuscate passwords* on page 77 for steps to obfuscate passwords.

If you want to use your own CA-signed certificates, you can import them in ABS.

Import existing CA-signed certificates

You can import your existing CA-signed certificate in ABS. To import the CA-signed certificate, stop ABS if it is already running. Complete the following steps to import the CA-signed certificate:

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>
   ```

   For 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 intermediate certificate from 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 by entering the following command. The command requires the destination keystore password. The destination keystore password entered in the command should be same 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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FF
   ```

   Enter the following command:

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

   For 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 `/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
```

Install MongoDB software

ABS uses a MongoDB database (4.2) to store analyzed logs and ABS cluster node information. MongoDB is installed using a replica set. In a replica set, MongoDB is installed on three nodes for high-availability (HA).

Update MongoDB default username and password

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:

```
db.createUser(
{
 user: "absuser",
 pwd: "abs123",
 roles: [ { role: "userAdminAnyDatabase", db: "admin" },
 { role: "readWrite", db: "abs_metadata" },
 { role: "readWrite", db: "abs_data" },
 { role: "readWrite", db: "abs_mldata" },
 { role: "readWrite", db: "local" } ]
});
```

Install MongoDB in replica set

Download either the RHEL 7 or Ubuntu 16 MongoDB 4.2 Linux `tarball` from the MongoDB website. For more information, see *https://www.mongodb.org/downloads*.

> ⓘ **Important:** This document describes a RHEL 7 download, but the equivalent Ubuntu version of MongoDB is also supported. Use the Ubuntu MongoDB URL to download the Ubuntu version.

Prerequisite:

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

> ⓘ **Important:** It is advised to follow MongoDB recommended setting, to avoid issues in your production MongoDB deployment. For more information, see *https://docs.mongodb.com/manual/administration/production-checklist-operations/* and *https://docs.mongodb.com/manual/administration/analyzing-mongodb-performance/*

Download MongoDB on three nodes which would form the replica set for high-availability (HA).

Install MongoDB one each node:

1. Create the MongoDB directory structure: create `mongo`, `data`, `logs`, and `key` directory on each MongoDB node.

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

2. Download MongoDB 4.2 on each node and extract to `/opt/pingidentity/mongo`

   ```
   # cd /opt/pingidentity/
   /opt/pingidentity# wget \
   https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.2.0.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. `absrs01` is the name of the replica set. 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 `abs_rs.js` file and edit to configure the IP address of the primary and secondary MongoDB nodes:

```
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
```

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

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

8. Verify that all the MongoDB nodes are running. On each MongoDB node, enter the following:

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

The primary node will display the following prompt:

```
absrs01:PRIMARY>
```

The secondary nodes will display the following prompt:

```
absrs01:SECONDARY>
```

9. Create User and initialize the database using `abs_init.js` file after making necessary modifications. You can set the following values in the file. However, ABS ships with default values

- Username and password
- Database names
- `training_period`
- `system_threshold_update_interval`
- `discovery_interval`
- `url_limit`
- `discovery_subpath`
- `api_discovery`
- `response_size`
- `enable_ssl`

On the primary node (node 1) Enter the following command:

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

ⓘ **Note:** user name and password should be changed from the default values.

The following is a snippet of the `abs_init.js` file:

```
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"
});
```

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 a self-signed certificate.

- **Using CA-signed certificate**: 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.

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

- **Using self-signed certificate**: To use a self-signed certificate then as a first-step generate a self-signed certificate and keys. 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
  ```

After either a CA-signed certificate or self-signed certificate has been added to the `key` directory, shut down MongoDB and restart with `--tlsMode` flag.

1. Shut down MongoDB:

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

2. Restart MongoDB with `-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
```

The --tlsMode flag can take the following three values:

- allowTLS
- preferTLS
- requireTLS

For more information on these options, see the *MongoDB documentation*.

Change default settings

It is recommended that you change the default key and password in ABS. Following is a list of commands to change the default values:

Change default JKS password

You can change the default password for KeyStore and the key. Complete the following steps to change the default passwords. Make sure that ABS is stopped before changing the JKS password.

> ⓘ **Important:** The KeyStore and Key password should be the same.

1. **Change the KeyStore password**: Enter the following command to change the KeyStore password. The default KeyStore password is `abs123`.

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

2. **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
```

Start ABS after you have changed the default passwords.

Change abs_master.key

Run the following command to create your own ABS master key to obfuscate keys and password in ABS.

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

Stop ABS: 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
```

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
```

Change CLI admin password

You can change the default admin password by entering 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
```

Change default access and secret key in MongoDB

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

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

1. Connect to MongoDB by entering the following command:

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

   `absuser` and `abs123` is the default user name 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>"} )
   ```

Connect ABS to MongoDB

Check and open MongoDB default port

The MongoDB default port for connection with ABS is 27017. Run the **`check_ports_abs.sh`** script on the ABS machine to determine whether the default port is available. Input the MongoDB host IP address and default port as arguments. For example:

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

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

Configure ABS to connect to MongoDB

ABS access key and secret key are used for MongoDB and REST API authentication. Edit `abs_init.js` in `/opt/pingidentity/mongo` directory to set the key values. Here is a sample `abs_init.js` file:

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

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

Copy the `abs_init.js` file from ABS

```
/opt/pingidentity/abs
        mongo
```

folder to the MongoDB system `/opt/pingidentity/mongo` folder.

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 4.2.0
connecting to: admin
switched to db abs_metadata
WriteResult({ "nInserted" : 1})
bye
```

Start and Stop ABS

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.

You can start ABS 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.

Start ABS as a service

Complete the following steps 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
   ```

Start ABS using start.sh script

To start ABS, run the **`start.sh`** script located in the `/opt/pingidentity/abs/bin` directory. Change working directory to `/opt/pingidentity/abs/bin`. Then start ABS by typing the following command:

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

To verify ABS has started, change working directory to `data` directory and look for two `.pid` files, `abs.pid` and `stream.pid`. 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%"
 } ] }
```

Stop ABS using stop.sh script

To stop ABS, first stop API Security Enforcer (if it is running) or turn OFF the ABS flag in API Security Enforcer. 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 using the `stop.sh` script, ABS would restart after stopping.

Stop ABS using service script

Run the following command to stop the ABS service: c

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

Part B – Install ASE

The ASE installation process is summarized below:

- Provision the system based on number of APIs and the expected queries per second (QPS). For information on sizing, contact PingIntelligence.
- Install ASE
- Configure ASE using the `/opt/pingidentity/ase/config/ase.conf` file
- Understand the ASE logical deployment options

ASE ports

ASE uses default ports as defined in the table below. If any ports configured in `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, 9090	ABS ports used by ASE for outbound connections to ABS for sending access logs and receive client identifiers of suspected attacks.

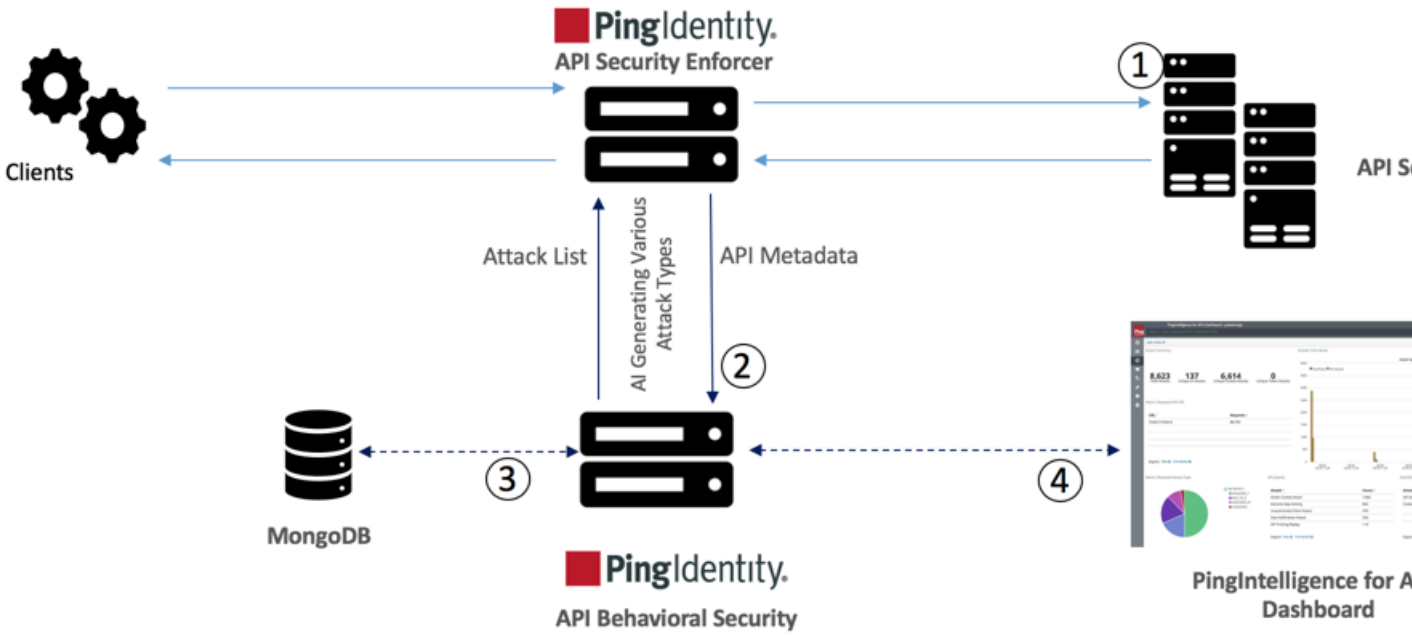
ⓘ **Important:** The management ports 8010 and 8020 should not be exposed to the Internet. If you are setting up the deployment in an AWS environment with security groups, use private IPs for ASE to ABS connections to avoid security group issues.

API Security Enforcer deployment modes

API Security Enforcer 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 balancers such as AWS ELB. In inline mode, API Security Enforcer deployed at the edge of the datacenter, terminates SSL connections from API clients. It then forwards the requests directly to the correct APIs – and app servers such as Node.js, WebLogic, Tomcat, PHP, etc.



API Security Enforcer
Inline Deployment Mode

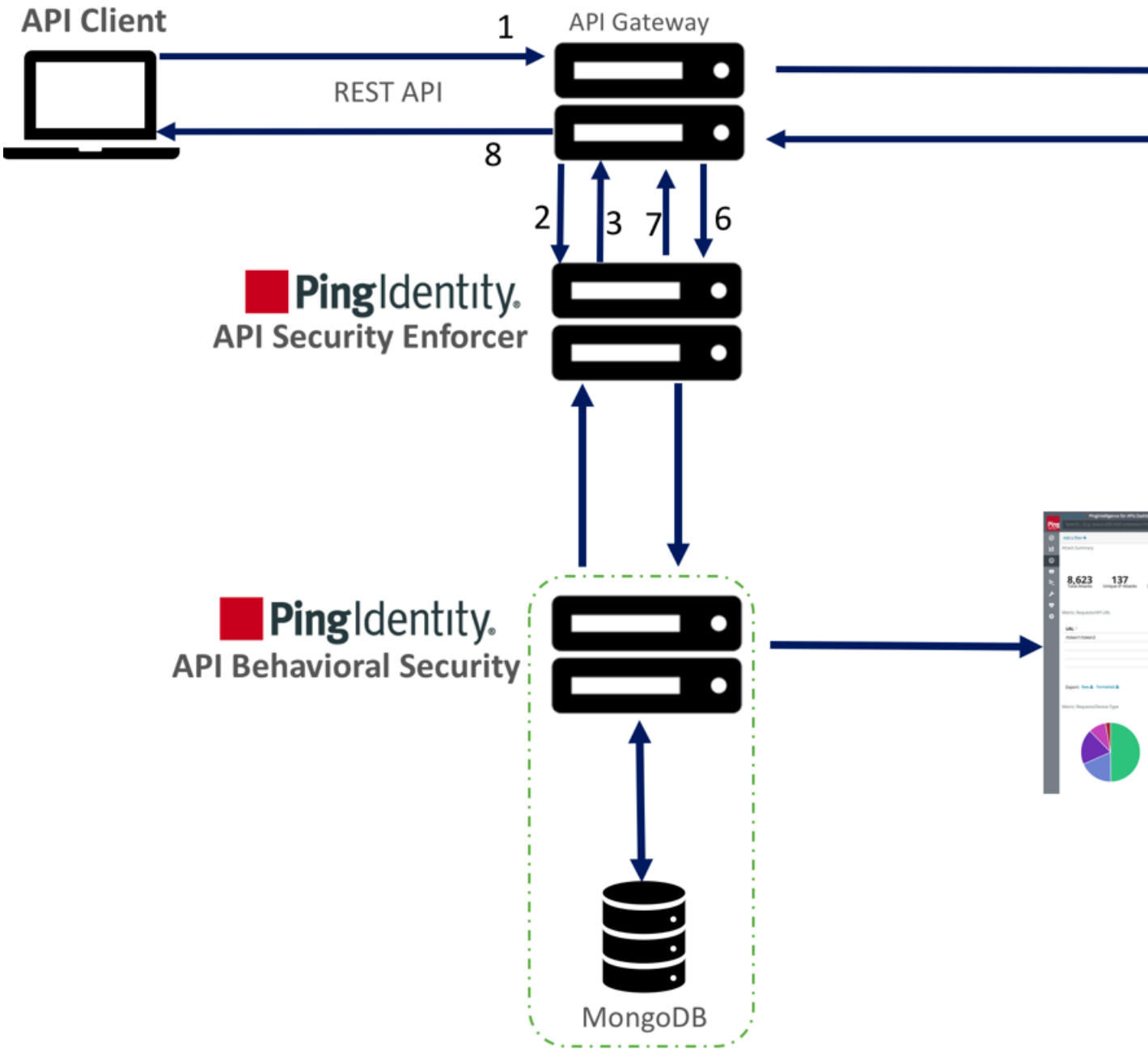
To configure ASE to work in the Inline mode, set the mode=inline in the `/opt/pingidentity/ase/config/ase.conf` file.

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

ASE when deployed in the sideband mode, 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 the Inline mode, set the mode=sideband in the `/opt/pingidentity/ase/config/ase.conf` file.



API Security Enforcer
Sideband Deployment Mode

Following is a description of the traffic flow through the API gateway and Ping Identity ASE.

1. Incoming request to API gateway
2. 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 Blacklist. If all checks pass, ASE returns a 200-OK response to the API gateway. Else, 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, else 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:** Complete the ASE sideband mode deployment by referring to API gateway specific deployment section on the *PingIdentity documentation site*.

Install ASE software

ASE supports RHEL 7.6 or Ubuntu 16.04 LTS on an EC2 instance, bare metal x86 server, and VMware ESXi.

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.

1. Go to the *download site*
2. Click on **Select** under PingIntelligence
3. Choose the correct build 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 working directory to `/opt` if you are installing the product as a root user. Choose any other location if you want to install ASE as a non-root user.
6. At the command prompt, type the following command to untar the ASE file:

   ```
   tar -zxvf <filename>
   ```

   For example:

   ```
   tar -zxvf pi-api-ase-rhel-4.2.tar.gz
   ```

7. To verify that ASE successfully installed, type the **`ls`** command at the command prompt. This should list the `pingidentity` directory and the build's `.tar` file. For example:

```
/opt/pingidentity/ase/bin/$ ls
pingidentity pi-api-ase-rhel-4.2.tar.gz
```

ASE license

To start 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.
- **Subscription license** – The subscription license is based on the subscription period. It is a good practice to *configure your email* before configuring the ASE license. ASE sends an email notification to the configured email ID in case the license has expired. Contact the PingIntelligence for APIs sales team for more information.

ⓘ **Note:** In case the subscription license has expired, ASE continues to run until a restart.

Configure ASE license

To configure the license in ASE, request for a license file from the PingIntelligence for APIs sales team. The name of the license file must be `PingIntelligence.lic`. Copy the license file to the `/opt/pingidentity/ase/config` directory and start ASE.

Update an existing license

If your existing license has expired, obtain a fresh license from PingIntelligence for APIs sales team and replace the license file in the `/opt/pingidentity/ase/config` directory. Make sure to stop and start ASE after the license file is updated.

Change default settings

It is recommended that you change the default key and password in ASE. Following is a list of commands to change the default values:

Change ase_master.key

Run the following command to create your own ASE master key to obfuscate keys and password in ASE.

Command: `generate_obfkey`. ASE must be stopped before creating a new `ase_master.key`

```
/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
```

Stop ASE: 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
```

Change ase_master.key: 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]:
```

Start ASE: 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
```

Change keystore password

You can change the keystore password by entering the following command. The default password is `asekeystore`. ASE must be running for updating the keystore password.

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
```

Change admin password

You can change the default admin password by entering the following command:

```
/opt/pingidentity/ase/bin/cli.sh update_password -u admin -p
Old password >
New password >
New password again >
Password updated successfully
```

Obfuscate keys and passwords

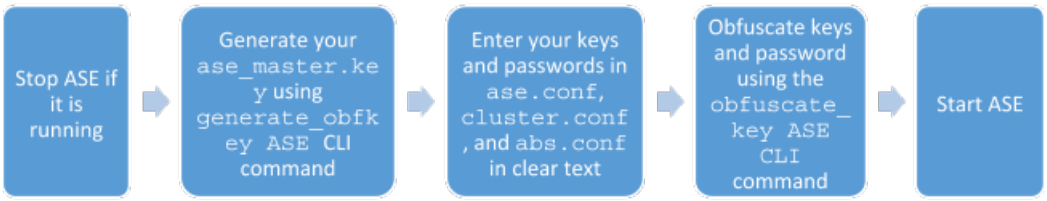
You must obfuscate the keys and passwords configured in `ase.conf`, `cluster.conf`, and `abs.conf` in the config directory. ASE ships with a default `ase_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `ase_master.key`.

The following keys and passwords are obfuscated in the three configuration files:

- `ase.conf` – Email and Keystore (PKCS#12) password
- `cluster.conf` – ABS access and secret key
- `abs.conf` – Cluster authentication key

ⓘ **Note:** During the process of obfuscation of keys and password, ASE must be *stopped*.

The following diagram summarizes the obfuscation process:



Generate your ase_master.key

You can generate the `ase_master.key` by running the `generate_obfkey` command in the ASE 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
```

The new `ase_master.key` is used to obfuscate the keys and passwords in the various configuration files.

ⓘ **Important:** In an ASE cluster, the new `ase_master.key` must be manually copied to each of the cluster nodes.

Obfuscate key and passwords

Enter the keys and passwords in clear text in `ase.conf`, `cluster.conf`, and `abs.conf`. Run the **`obfuscate_keys`** command to obfuscate keys and passwords:

```
/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
```

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.

Tune host system for high performance

ASE ships with a script to tune the host Linux operating system for handling high TCP concurrency and optimizing performance. To understand the tuning parameters, refer to 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

ⓘ **Important:** 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 7.6. For tuning Ubuntu 16.04 LTS, use the Ubuntu tuning scripts.

Tune the host system:

Enter the following command in the command line:

```
/opt/pingidentity/ase/bin/tune_rhel7.sh
```

Make sure to close the current shell after running the tune script and proceeding to start ASE.

ⓘ **Note:** If ASE is deployed in a Docker Container, run the tune script on the host system, not in the container.

Untune the host system:

The “untune” script brings the system back to its original state. Enter the following command in the command line:

```
/opt/pingidentity/ase/bin/untune_rhel7.sh
```

ⓘ **Note:** You should be a `root` user to run the tune and untune scripts.

Start and Stop ASE

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 `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
```

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.

Start ASE as a service

Complete the following steps 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
   ```

Start ASE using start.sh script

Change working directory to `bin` and run the **`start.sh`** script.

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 4.2...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Stop ASE using stop.sh

Change working directory to `bin` and 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
```

Stop ASE using service script

Run the following command to stop the ASE service:

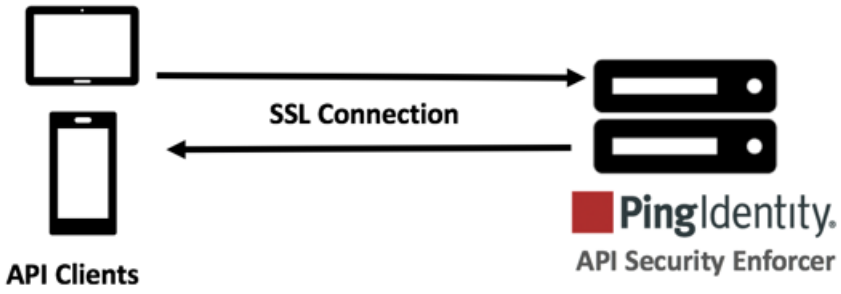
```
systemctl stop pi-ase.service
```

Configure SSL for external APIs

ASE supports both TLS 1.2 and SSLv3 for external APIs. You can configure SSL in ASE for client side connection using one of the following methods:

- **Method 1**: Using CA-signed certificate
- **Method 2**: Using self-signed certificate
- **Method 3**: Importing 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 illustration below:



In a cluster setup:

1. Stop all the ASE cluster nodes
2. Configure the certificate on the management node. 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

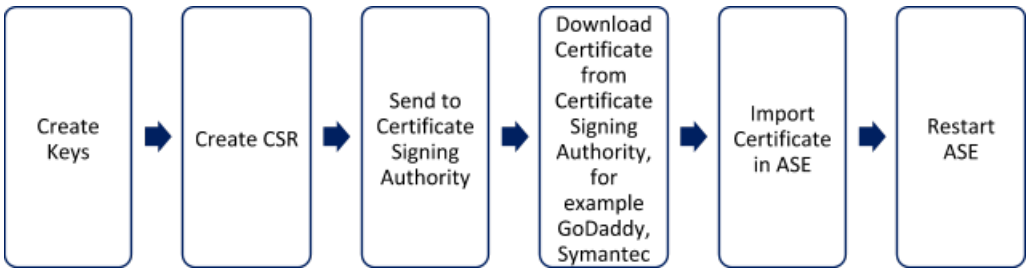
Enable SSLv3

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
```

Method 1: Using 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:



ⓘ **Note:** ASE internally validates the authenticity of the imported certificate.

To use a CA-signed certificate:

1. Create a private key. ASE CLI is used to create a 2048-bit private key and to store it in the keystore.

```
/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.

```
/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 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 keystore.

```
/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.

Method 2: Use self-signed certificate

A self-signed certificate is also supported for customer testing.

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 `/opt/pingidentity/ase/config/certs/dataplane/dh1024.pem` directory.

```
/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 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.

Method 3: import 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. 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 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.

ⓘ **Important:** You can also configure for Management APIs. For more information on configuring SSL for management APIs, see *Configure SSL for Management APIs*.

ASE cluster setup (optional)

For production environments, Ping Identity recommends setting up a cluster of ASE nodes for improved performance and availability.

ⓘ **Note:** Enable NTP on each ASE node system. All cluster nodes must be in the same time zone.

To setup an ASE cluster node:

1. Navigate to the `config` directory
2. Edit `ase.conf` file:
   a. Set `enable_cluster=true` for all cluster nodes.
   b. Confirm that the parameter mode is the same on each ASE cluster node, either inline or sideband. If parameter mode values do not match, the nodes will not form a cluster.

3. Edit the `cluster.conf` file:
   a. Configure cluster_id with an identical value for all nodes in a single cluster (for example, `cluster_id=shopping`)
   b. Enter port number in the cluster_manager_port parameter. ASE node uses this port number to communicate with other nodes in the cluster.
   c. 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.
   d. 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.

   Here 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 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
```

Scale up the ASE cluster

Scale up the ASE cluster by adding nodes to an active cluster without disrupting traffic. To add a new cluster node, enter the `peer_node` IP address or hostname in the `cluster.conf` file of the ASE node and then *start the ASE node*. The new node will synchronize configuration and cookie data from the peer nodes. After loading, it will become part of the cluster. For 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
```

Scale down the ASE cluster

A node can be removed from an active cluster without disrupting traffic by performing the following:

1. Stop the ASE node to be removed.
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.

Delete a cluster node

An inactive cluster node has either become unreachable or has been stopped. When you delete a stopped cluster node, the operation does not remove cookie and other synchronized data. To find which cluster nodes are inactive, use the **`cluster_info`** command:

```
/opt/pingidentity/ase/bin/cli.sh cluster_info -u admin -p
cluster id : ase_cluster
cluster nodes
127.0.0.1:8020 active
1.1.1.1:8020 active
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
```

Using the **`cluster_info`** command output, you can remove the inactive cluster nodes 2.2.2.2:8020 and 172.17.0.5:8020.

To delete the inactive node, use the **`delete_cluster_node`** command:

```
/opt/pingidentity/ase/bin/cli.sh delete_cluster_node <IP:Port>
```

Stop ASE cluster

Stop the entire cluster by running the following command on any node in the cluster.

```
/opt/pingidentity/ase/bin/stop.sh cluster –u admin –p
```

When the cluster stops, each cluster node retains all the cookie and certificate data.

Part C – Integrate ASE and ABS

The ABS Engine installation process is summarized below:

- Connect ASE to ABS AI engine for ASE to send access log files to ABS.
- Enable ASE to ABS engine communication: Just connecting ASE and ABS engine does not mean that access logs would be sent by ASE to ABS. ASE to ABS communication has to be enabled separately.
- Add API JSON files to ASE. The API JSON files define your API and its various parameters. For more information, see *Defining an API JSON* file.
- ABS AI engine models need to be trained for it to analyze and report on your API traffic.

Connect ASE to ABS AI engine

Check ABS port availability

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_ase.sh {ABS IPv4:[port]}
```

Configure ASE

Update `abs.conf` located in the ASE `/opt/pingidentity/ase/config` directory with ABS Engine address and authentication keys:

- Configure abs_endpoint with the ABS Engine management IP address / host name and port number (Default: 8080) which was configured in the `/opt/pingidentity/abs/config/abs.properties` file.

  ⓘ **Note:** Note: If ABS is in a different AWS security group, use a private IP address

- Configure ABS access_key and secret_key using the key values from the `abs_init.js` file located in `/opt/pingidentity/abs/mongo`.

Here 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.

; 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.

Enable ASE to ABS engine communication

To start communication between ASE and the AI engine, run the following command:

```
./cli.sh enable_abs –u admin -p admin
```

To confirm an ASE Node is communicating with ABS, issue the ASE status command:

```
/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
```

Add APIs to ASE

After you enable ASE to ABS communication, add APIs to ASE. Read the following topics to define and add APIs to ASE:

- *API naming guidelines* on page 153
- *Defining an API – API JSON configuration file* on page 153

Train ABS AI engine

For ABS to start predicting various attacks types, the model needs to be trained. The number of hours (default - 24 hours) is configurable for model training. Set the value of training_period parameter in the `abs_init.js` file in the `/opt/pingidentity/mongo` directory. For more detailed information about training AI model, see *AI Engine training* on page 312.

```
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": true,
 "discovery_initial_period" : "1",
 "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"
});
```

Start the training

The training starts as soon as ABS receives the first API traffic from API Security Enforcer and continues for the number of hours set in the attack_initial_training parameter. Training occurs automatically when a new API is added.

Verify training completion

ABS training status is checked using the ABS Admin API which returns the training duration and prediction mode. 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. The API URL for Admin API is: https://*<ip>:<port>*/v4/abs/admin. 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
```

IP and port number is of the ABS machine.

ⓘ **Note:** ABS only detects attacks after the training period is over. During training, no attacks are generated.

Part D – Install PingIntelligence Dashboard

Installing PingIntelligence for APIs Dashboard automatically installs Elasticsearch, Kibana To install PingIntelligence Dashboard, ensure that the following prerequisites are met:

- **Server**: 8 core CPU, 16 GB, 1 TB HDD
- **Operating system**: RHEL 7.6 or Ubuntu 16.0.4 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
- **ABS**: ABS URL, access, and secret key. Make sure that ABS is reachable from the PingIntelligence Dashboard machine.
- **ASE**: ASE management URL, access, and secret key. Make sure that ASE is reachable from the PingIntelligence Dashboard machine.

Port numbers

The following is a list of default port numbers. Make sure that these are available for installing PingIntelligence Dashboard.

- **PingIntelligence Dashboard**: 8030
- **Elasticsearch**: 9200
- **Kibana**: 5601
- **H2 database**: 9092. H2 database is installed and runs as a part of PingIntelligence Dashboard.

Supported browsers: The following Web browsers are supported:

- Google Chrome: Version 49 or later
- Mozilla Firefox: Version 69 or later
- Microsoft Edge: Version 42 or later
- Apple Safari: Version 11.1 or later

Operating system configurations: Complete 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
  ```

- **JDK installation**: Set `JAVA_HOME` to `<jdk_install>` directory and add `<jdk_install>/bin` to system `PATH` variable

- Choose the `<pi_install_dir>` directory. `<pi_install_dir>` should be readable and writable by the logged in user.

PingIntelligence Dashboard users

There are two pre-configured login users in PingIntelligence Dashboard. The two users are:

- `admin`
- `ping_user`

Multiple `admin` and `ping_user` can simultaneously log into PingIntelligence Dashboard. The admin user has full access to PingIntelligence 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 log in simultaneously.

Install PingIntelligence Dashboard

Complete the following steps to install PingIntelligence Dashboard:

1. Create a `ping_install_dir` directory on your host machine
2. *Download* the PingIntelligence Dashboard binary
3. *Download* Elasticsearch 6.8.1
4. *Download* Kibana 6.8.1
5. Change directory to `ping_install_dir`:

```
# cd pi_install_dir
```

6. Untar the PingIntelligence Dashboard:

```
# tar -zxf pi-api-dashboard-4.1.1.tar.gz
```

7. Change directory to `/pingidentity/webgui/`

```
# cd pingidentity/webgui/
```

8. Install PingIntelligence Dashboard by entering the following command and follow the instructions displayed on the prompt:

```
# ./bin/pi-install-ui.sh
```

```


elasticsearch-6.8.1.tar.gz file path >
kibana-6.8.1-linux-x86_64.tar.gz file path >

Use bundled ssl key and self signed certificate for ui server [y/n]? >[n]
ssl private key path >
ssl certificate path >

Use default password [changeme] for all components and users [y/n]? > [n]
UI login admin user 'admin' password >
Renter UI login admin user 'admin' password >
UI login regular user 'ping_user' password >
Renter UI login regular user 'ping_user' password >

ABS url >
Use default access/secret key for ABS  [y/n] ? > [n]
ABS access key >
ABS secret key >

ASE management url >
Use default access/secret key for ASE  [y/n] ? > [n]
ASE access key >
```

```
ASE secret key >


configuring elasticsearch… please wait for 15 seconds
elasticsearch config is completed.

configuring kibana…please wait 60 seconds
kibana config is completed.

configuring dashboard…
generating new obfuscation master key
dashboard config is completed.

configuring webgui…
generating new obfuscation master key
webgui config is completed.

saving auto generated credentials for all components to
 webgui_internal.creds file

WebGUI installation completed.

Start WebGUI [y/n] > [y]

start elasticsearch...
 elasticsearch started. Log is available at elasticsearch/logs/
elasticsearch.log

start dashboard……
 dashboard started. Log available at dashboard/logs/dashboard.log

start kibana……
  kibana started. Log available at kibana/logs/kibana.log

start ui server……
  UI server started. Log available at webgui/logs/admin/admin.log

WebGUI started. Log available at webgui/logs/admin/admin.log

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.

Important Action:
1) Credentials for all internal components are available in
 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) Two obfuscation master keys are auto-generated
   pingidentity/webgui/config/webgui_master.key
   pingidentity/dashboard/config/dashboard_master.key
3) For security purposes you should move obfuscation master keys from this
 server. But when components
   are restarted, master keys should be present at the original locations.
```

Verify the installation

You can verify the installation by checking the process IDs (PID) of each component. You can check the `pid` of components at the following location:

- **Elasticsearch**: `<pi_install_dir>/elasticsearch/logs/elasticsearch.pid`
- **Kibana**: `<pi_install_dir>/kibana/logs/kibana.pid`
- **Dashboard**: `<pi_install_dir>/dashboard/logs/dashboard.pid`

Tune Dashboard performance parameters

Configure the following three parameters for Dashboard's better performance.

Parameter	Description	Location
Elasticsearch		
-Xms and -Xmx	▪ **Xms** - Defines the minimum heap size of Elasticsearch. Set it to 4GB as `Xms4g`. ▪ **Xmx** - Defines the maximum heap size of Elasticsearch. Set it to 4GB as `Xmx4g`.	`$ES_HOME/config/jvm.options`
thread_pool.search.size	Defines thread pool size for count/search/suggest operations in Elasticsearch. Configure it to 50% of total CPUs allocated.	`$ES_HOME/config/elasticsearch.yml`
Kibana		
elasticsearch.requestTimeout	Time (in milliseconds) to wait for Elasticsearch to complete the request and return the response back to Kibana. Set the value to 60000 milliseconds.	`$kibana_HOME/config/kibana.yml`

ⓘ **Note:** To detect and mitigate attacks like Cross Site Scripting(XSS), 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;
```

Start and stop Dashboard

You can choose to start and stop all the components together or individually. 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.. [started]

Verifying elasticsearch connectivity.. [OK]
Verifying ABS connectivity.. [OK]

Starting dashboard.. [started]
Starting kibana.. [started]


Verifying Kibana connectivity.. [OK]
```

```
Verifying ASE connectivity.. [OK]

Starting webgui.. [started]

WebGUI started.
```

To stop all the components of PingIntelligence Dashboard together, enter the following command:

```
# cd <pi_install_dir>/pingidentity/webgui
# ./bin/stop-all.sh

Stopping webgui..[stopped]
Stopping dashboard.. [stopped]
Stopping  kibana.. [stopped]
Stopping  elasticsearch.. [stopped]

WebGUI stopped.
```

Start and stop PingIntelligence Dashboard components individually

Start the components in the following order:

1. **Start Elasticsearch**: Enter the following command to start Elasticsearch:

   ```
   # cd <pi_install_dir>/pingidentity/elasticsearch
   # ./bin/elasticsearch -d -p logs/elasticsearch.pid
   ```

2. **Start Dashboard**: Enter the following command to start Dashboard:

   ```
   # cd <pi_install_dir>/pingidentity/dashboard
   # ./bin/start.sh
   ```

3. **Start Kibana**: Enter the following command to start Kibana:

   ```
   # cd <pi_install_dir>/pingidentity/kibana
   # ./bin/kibana >> ./logs/kibana.log 2>&1 & echo $! > logs/kibana.pid
   ```

4. **Start Web GUI**: Enter the following command to start Web GUI:

   ```
   # cd <pi_install_dir>/pingidentity/webgui
   # ./bin/start.sh
   ```

Stop the components individually by entering the following commands:

Stop Elasticsearch: Stop Elasticsearch by entering the following command:

```
# cd <pi_install_dir>/pingidentity/elasticsearch
# kill -15 "$(<logs/elasticsearch.pid)"
```

Stop dashboard engine: Stop the dashboard engine by entering the following command:

```
# cd <pi_install_dir>/pingidentity/dashboard
# ./bin/stop.sh
```

Stop Kibana: Stop Kibana by entering the following command:

```
# cd <pi_install_dir>/pingidentity/kibana
# kill -9 "$(<logs/kibana.pid)"
```

Stop Web GUI: Enter the following command to stop Web GUI:

```
# cd <pi_install_dir>/pingidentity/webgui
# ./bin/stop.sh
```

Start and stop PingIntelligence Dashboard as a service

You can also start and stop PingIntelligence Dashboard as a service. Complete the following steps to start PingIntelligence Dashboard as a service:

1. Navigate to the `util` directory and run the following command to install PingIntelligence Dashboard as a service:

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

2. Start the service by entering the following command:

   ```
   systemctl start pi-webgui.service
   ```

To stop PingIntelligence Dashboard service, run the following command:

```
systemctl stop pi-webgui.service
```

Part E – Access 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.

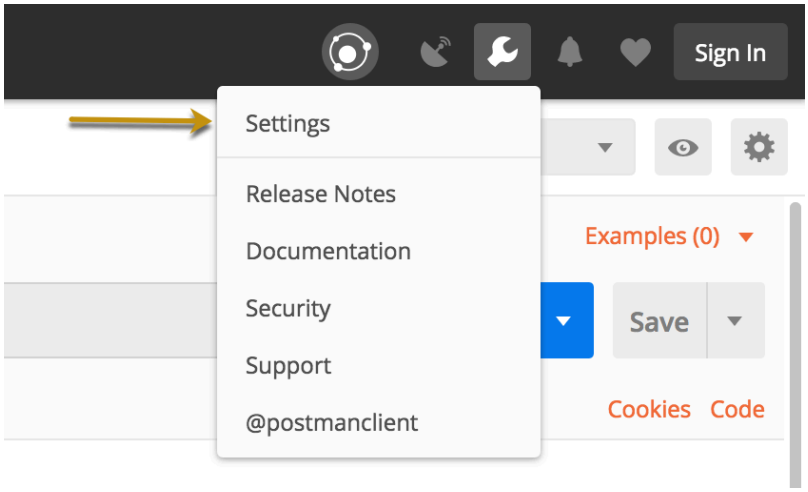
Install Postman with PingIntelligence for APIs Reports

Ping Identity provides configuration files which are used by *Postman* to access the ABS REST API JSON information reports. Make sure to install Postman 6.2.5 or higher.

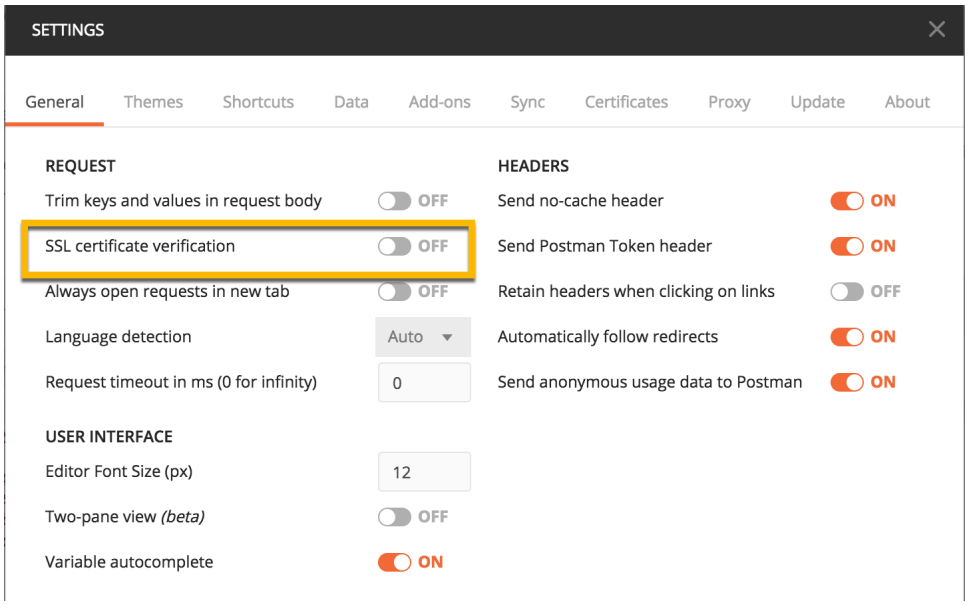
Using ABS self-signed certificate with Postman

ABS ships with a self-signed certificate. If you want to use Postman with the self-signed certificate of ABS, then from Postman's settings, disable the certificate verification option. Complete the following steps to disable Postman from certificate verification:

1. Click on the **spanner** on the top-right corner of Postman client. A drop-down window is displayed.
2. Select **Settings** from the drop-down window:



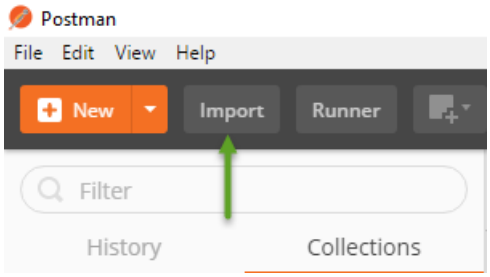
3. In the Settings window, switch-off certificate verification by clicking on the SSL certificate verification button:



View ABS Reports in Postman

To view the reports, complete the following steps:

1. Download ABS_Environment and ABS_Reports JSON files from **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.



4. After importing the files, click the gear  button in the upper right corner.
5. In the **MANAGE ENVIRONMENTS** pop-up window, click **ABS_4.1_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 the `opt/pingidentity/mongo/abs_init.js` for either 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 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.
   - **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.

   ⓘ **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**. ABS external REST APIs section provides detailed information on each API call and the JSON report response.

Part F - 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*

After you have completed the parts A to E of 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* on page 489
- *PingIntelligence Apigee Integration* on page 513
- *PingIntelligence AWS API Gateway Integration* on page 531
- *Azure APIM sideband integration* on page 573
- *Axway sideband integration* on page 549
- *PingIntelligence - CA API gateway sideband integration* on page 582
- *F5 BIG-IP PingIntelligence integration* on page 591
- *IBM DataPower Gateway sideband integration* on page 603
- *PingIntelligence - Kong API gateway integration* on page 610
- *Mulesoft sideband integration* on page 616
- *NGINX sideband integration* on page 631
- *NGINX Plus sideband integration* on page 646
- *PingAccess sideband integration* on page 665
- *PingIntelligence WSO2 integration* on page 677

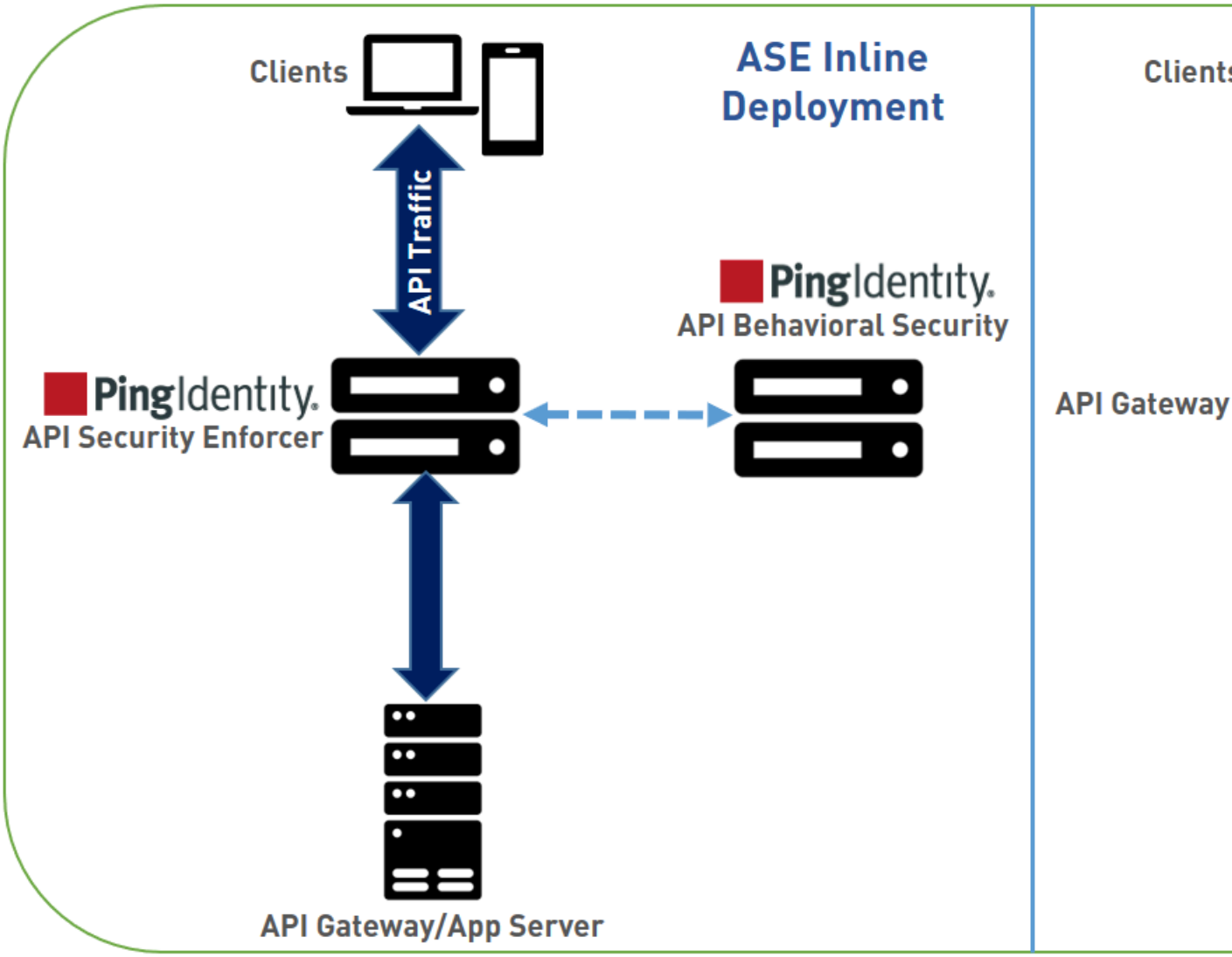
API Security Enforcer

Introduction

ASE supports multiple deployments modes to provide customers flexibility in deploying PingIntelligence for APIs API cybersecurity. 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 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 table shows a summary of features available in each deployment options.

Security Features	Inline	Sideband
Interface to ABS AI Engine for AI powered attack detection	Yes	Yes
API deception – 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, usernames, cookies, and IP addresses.	Yes	Yes
Black and whitelist 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	
Simple deployment with modular JSON configuration files	Yes	Yes
Live updates – Add/remove without loss of traffic or stopping services.	Yes	Yes
Obfuscation – Keys and passwords are obfuscated	Yes	Yes
Active-active clustering – 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
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

Administration

API Security Enforcer (ASE) is deployed by modifying configuration files to support your environment. The configuration files consist of the following:

- `ase.conf` – the master configuration file with parameters to govern ASE functionality.
- `cluster.conf` – configures ASE cluster setup.
- `abs.conf` – configures ASE to ABS (AI Engine) connectivity. ASE sends log files to ABS for processing and receives back client identifiers (for example, token, IP address, cookie) to block.

ASE license

To start 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.
- **Subscription license** – The subscription license is based on the subscription period. It is a good practice to *configure your email* before configuring the ASE license. ASE sends an email notification to the configured email ID in case the license has expired. Contact the PingIntelligence for APIs sales team for more information.

> ⓘ **Note:** In case the subscription license has expired, ASE continues to run until a restart.

Configure ASE license

To configure the license in ASE, request for a license file for the PingIntelligence from APIs sales team. The name of the license file must be `PingIntelligence.lic`. Copy the license file to the `/opt/pingidentity/ase/config` directory and start ASE.

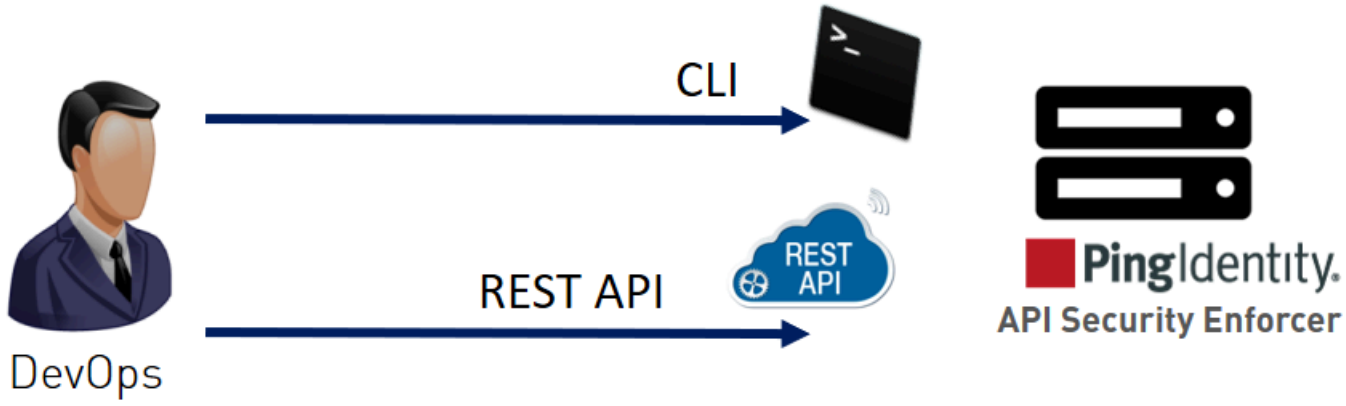
Update an existing license

If your existing license has expired, obtain a fresh license from PingIntelligence for APIs sales team and replace the license file in the `/opt/pingidentity/ase/config` directory. Make sure to stop and start ASE after the license file is updated.

ASE interfaces

The interfaces to configure and operate ASE consist of:

- Command line interface (CLI)
- ASE REST API



ASE CLI

Located in the `bin` directory, `cli.sh` is the script that administers ASE and performs all ASE functions except starting and stopping ASE. To execute commands, type **`cli.sh`** followed by the command name. To see a list of all commands, type the following command at the CLI:

`/opt/pingidentity/ase/bin/cli.sh`

The following table lists some basic CLI commands. For a complete list, see *CLI for inline ASE* on page 234 and *CLI for sideband ASE* on page 177

Option	Description
`help`	Displays **`cli.sh`** help
`version`	Displays ASE's version number
`status`	Displays ASE's status.
`update_password`	Updates the password for ASE admin account.

ⓘ **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, follow *Updating a Configured API*

CLI commands include the following:

help command

To get a list of CLI commands, enter the help command:

```
/opt/pingidentity/ase/bin/cli.sh help
```

version command

```
To query system information, enter the version command:
/opt/pingidentity/ase/bin/cli.sh version
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
```

status command

To get ASE `status`, enter the status command:

```
/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, 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
```

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 backend servers, the number of APIs, and so on.

ASE REST API commands consist of the following:

- **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)
- **Flow Control:** Read flow control (GET), Update flow control for API (POST), Update flow control of a Server for an API (POST)

Customizing ASE ports

ASE uses default ports as defined in the table below. If any port configured in `ase.conf` file is unavailable, ASE will not start.

Port Number	Usage
80	Data port. HTTP and WebSocket (ws) connections. If you are installing ASE as a non-root user, then use port greater than 1024.
443	Data port. HTTPS and Secure WebSocket (wss) connections. If you are installing ASE as a non-root user, then use port greater than 1024.
8010	Management port. Used by CLI and REST API for managing ASE.
8020	Cluster port. Used by ASE internally to set up the cluster.
8080, 9090	ABS ports. 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 AWS environment, both management ports should be private in the Security Group for ASE.

Security Group “ase”:

`port 80`: Accessible from any client (note: not secure)

`port 443`: Accessible from any client

`port 8010`: Accessible from management systems and administrators

`port 8020`: Accessible from peer ASE nodes

> ⓘ **Note:** If you are setting up the deployment in an AWS environment with security groups, use private IPs for ABS connections to avoid security group issues.

Configure time zone

You can set up ASE in either `local` or `UTC` time zone by configuring the timezone parameter in `/pingidentity/ase/config/ase.conf` file. All 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 by default runs in the local time zone. Following is a snippet of `ase.conf` for timezone parameter.

```
; Set the timezone to utc or local. The default timezone is local.
timezone=local

<truncated ase.conf...>
```

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. However, if you have used manual installation, then you need to manually configure the time zone on each ASE node.

You can use ASE **`status`** command to check the current time zone of ASE.

```
#./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)
```

Change ASE time zone

If you want to change the time zone in ASE, complete the following steps:

1. Stop ASE
2. Update the timezone parameter in `ase.conf` file
3. Start ASE

Tune host system for high performance

ASE ships with a script to tune the host Linux operating system for handling high TCP concurrency and optimizing performance. To understand the tuning parameters, refer to 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

> ⓘ **Important:** 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 7.6. For tuning Ubuntu 16.04 LTS, use the Ubuntu tuning scripts.

Tune the host system:

Enter the following command in the command line:

```
/opt/pingidentity/ase/bin/tune_rhel7.sh
```

Make sure to close the current shell after running the tune script and proceeding to start ASE.

ⓘ **Note:** If ASE is deployed in a Docker Container, run the tune script on the host system, not in the container.

Untune the host system:

The “untune” script brings the system back to its original state. Enter the following command in the command line:

```
/opt/pingidentity/ase/bin/untune_rhel7.sh
```

ⓘ **Note:** You should be a `root` user to run the tune and untune scripts.

Start and stop ASE

Prerequisite:

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. You can run ASE as a non-root user also.

Start ASE

Before starting ASE, make sure that `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
```

Change working directory to `bin` and run the `start.sh` script.

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 4.0.2...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Stop ASE

Change working directory to `bin` and 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
```

Change default settings

It is recommended that you change the default key and password in ASE. Following is a list of commands to change the default values:

Change ase_master.key

Run the following command to create your own ASE master key to obfuscate keys and password in ASE.

Command: `generate_obfkey`. ASE must be stopped before creating a new `ase_master.key`

```
/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
```

Stop ASE: 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
```

Change ase_master.key: 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]:
```

After you change the `ase_master.key`, you need 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 plain text and run the obfuscation commands. For more information on obfuscation, see *Obfuscate keys and passwords* on page 119.

Start ASE: 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 4.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Change keystore password

You can change the keystore password by entering the following command. The default password is `asekeystore`. ASE must be running for updating the keystore password.

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
```

Change admin password

You can change the default admin password by entering the following command:

```
/opt/pingidentity/ase/bin/cli.sh update_password -u admin
Old password >
New password >
New password again >
```

```
Password updated successfully
```

ⓘ **Note:** Any change in the ASE admin password must be updated in the PingIntelligence for APIs Dashboard. Add the new password to `<pi_install_dir>/webgui/config/webgui.properties` file and obfuscate it.

Obfuscate keys and passwords

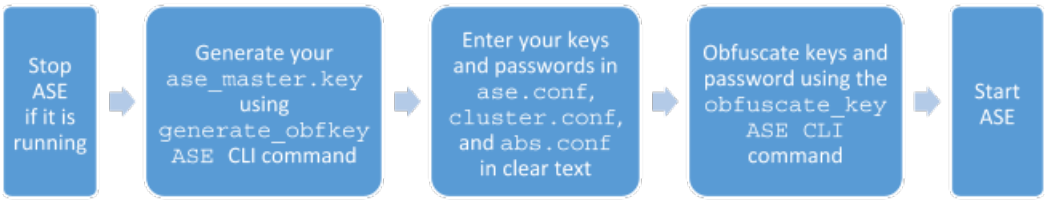
Using the ASE command line interface, you can obfuscate keys and passwords configured in `ase.conf`, `cluster.conf`, and `abs.conf`. Here is the obfuscated data in each file:

- `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. It is recommended to generate your own `ase_master.key`.

ⓘ **Note:** During the process of obfuscation password, ASE must be stopped.

The following diagram summarizes the obfuscation process:



Generating your `ase_master.key`

You can generate the `ase_master.key` by running the **`generate_obfkey`** ASE CLI command.

```
/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
```

The new `ase_master.key` is used to obfuscate the keys and passwords in the configuration files.

ⓘ **Important:** In an ASE cluster, the `ase_master.key` must be manually copied to each cluster node.

Obfuscate keys and passwords

Enter the keys and passwords in clear text in `ase.conf`, `cluster.conf`, and `abs.conf`. Run the `obfuscate_keys` command to obfuscate keys and passwords:

```
/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
```

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.

PKCS#12 keystore

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. You must update the default PKCS#12 keystore password by using the **`update_keystore_password`** command for security reasons. The password is updated and obfuscated at the same time. ASE must be running for updating the keystore password.

```
/opt/pingidentity/ase/bin/cli.sh update_keystore_password -u admin -p admin
New password >
New password again >
keystore password updated
```

Directory structure

During the installation process, ASE creates the following directories:

Directory Name	Purpose
`config`	Contains files and directories to configure ASE and its APIs. The `certs` subdirectory contains the keys and certificates for 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 ABS for analysis. The access log files are compressed and moved to `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

ASE cluster setup

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`, API JSON files) synchronization across all cluster nodes
- Update and delete operations using CLI and REST APIs
- Run time addition or deletion of cluster nodes
- Real-time blacklist synchronization across cluster
- A single cluster with nodes spanning across multiple data centers

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.

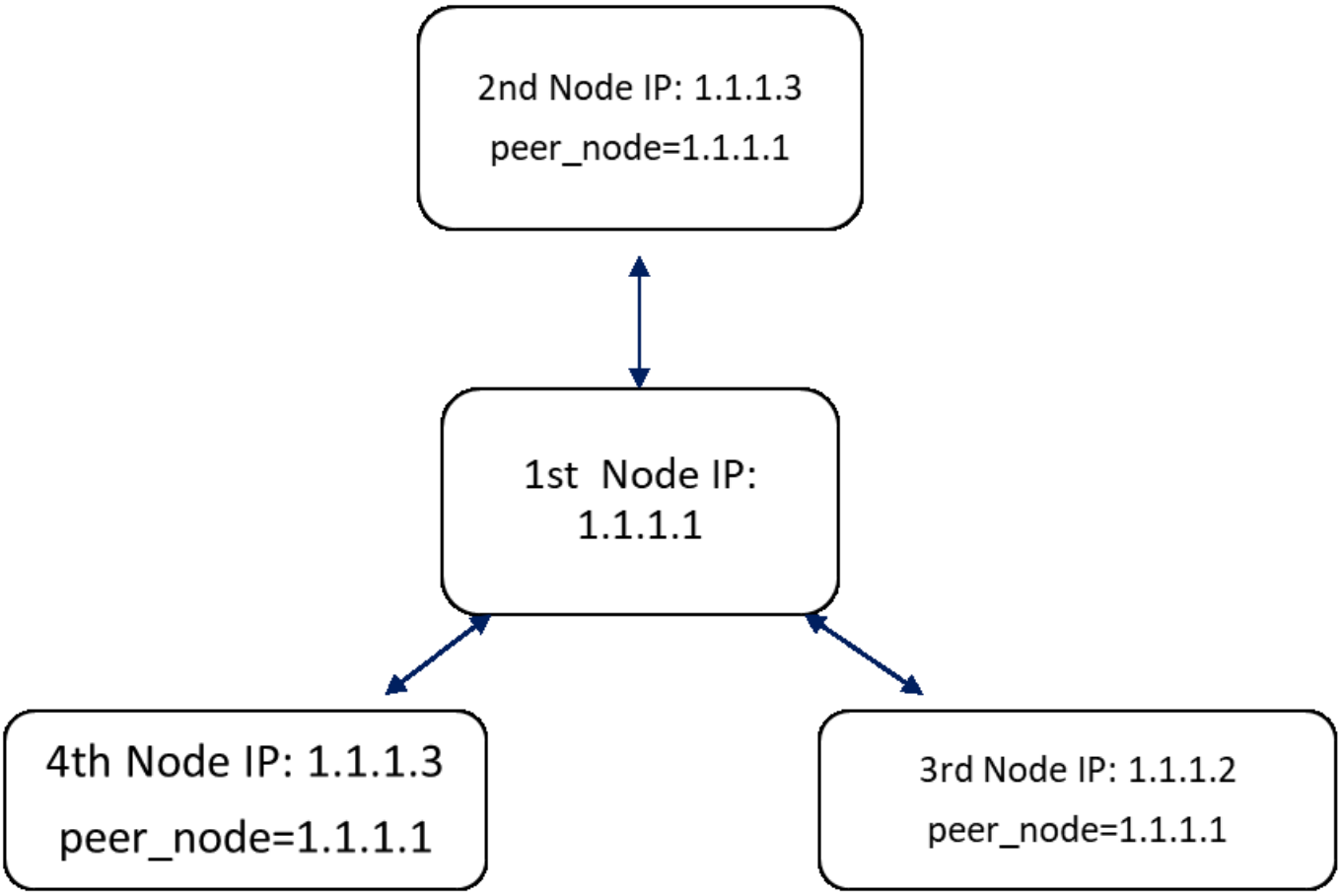
ASE cluster deployment

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` 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 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.



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
Cookies	Yes	Live and restart
CLI admin password	No	No
Authorization token for sideband ASE	Yes	Live and restart
Blacklist and whitelist (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.	-

Start ASE cluster

To setup an ASE cluster, the following three steps must be completed:

Pre-requisites

1. Obtain list of IP addresses and ports required for ASE cluster nodes
2. Enable NTP on your system.

3. If adding an existing ASE instance to a cluster, backup the ASE data first. When a node is added to a cluster, it synchronizes the data from the other nodes and overwrites existing data.

To setup an ASE cluster node:

1. Navigate to the `config` directory
2. Edit `ase.conf` file:
   a. Set `enable_cluster=true` for all cluster nodes.
   b. Make sure that the value in the parameter mode is same on each ASE cluster node, either `inline` or `sideband`. If the value of mode parameter does not match, the nodes will not form a cluster.
3. Edit the `cluster.conf` file
   a. Configure `cluster_id` with an identical value for all nodes in a single cluster (for example, `cluster_id=shopping`)
   b. Enter port number in the cluster_management_port (default port is 8020) parameter. ASE node uses this port number to communicate with other nodes in the cluster.
   c. Enter an IPv4 address or hostname with the port number for the peer_node which is the first (or any existing) node in the cluster. Keep this parameter empty for the first node of the cluster.
   d. Provide the obfuscated `cluster_secret_key`. All the nodes of the cluster must have the same obfuscated `cluster_secret_key`. This key must be entered manually on each node of the cluster for the nodes to connect to each other.
   e. 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 hostname of the first cluster in the node in the `peer_node` variable.

Here 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:7LFcqXQlqkW9kldQoFg0nJoLSojnzHDbD3
```

After configuring an ASE node, start the node by running the following command:

```
/opt/pingidentity/ase/bin/start.sh
```

Scale up the ASE cluster

Scale up the ASE cluster by adding one node at a time to an active cluster without disrupting traffic. To add a new cluster node, enter the peer_node IP address or hostname in the `cluster.conf` file of the ASE node and then start the ASE node. The new node will synchronize configuration and cookie data from the peer nodes. After loading, it will become part of the cluster. For 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
```

Scale down ASE cluster

A node can be removed from an active cluster without disrupting traffic by completing the following stops:

1. Stop the ASE node to be removed using the *`stop` 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

Delete ASE cluster node

An inactive cluster node has either become unreachable or has been stopped. When you delete a stopped cluster node, the operation does not remove cookie and other synchronized data. To find which cluster nodes are inactive, use the **`cluster_info`** command:

```
/opt/pingidentity/ase/bin/cli.sh cluster_info -u admin -p
cluster id : ase_cluster
cluster nodes
127.0.0.1:8020 active
1.1.1.1:8020 active
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
```

Using the **`cluster_info`** command output, you can remove the inactive cluster nodes 2.2.2.2:8020 and 172.17.0.5:8020.

To delete the inactive node, use the **`delete_cluster_node`** command:

```
/opt/pingidentity/ase/bin/cli.sh delete_cluster_node <IP:Port>
```

Stop ASE cluster

You can stop the entire cluster by running the following command on any ASE node in the cluster.

```
/opt/pingidentity/ase/bin/stop.sh cluster –u admin –p
```

When the cluster stops, each cluster node retains all the cookie and certificate data.

Restart ASE cluster

ASE cluster nodes must be restarted, one node at a time, to ensure consistent cluster behavior. To restart the ASE Cluster, complete the following 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 executing 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. Start the next node in the cluster only after the status of the node changes to `started`.

```
/opt/pingidentity/ase/bin/cli.sh status -u admin -p
Status: started
```

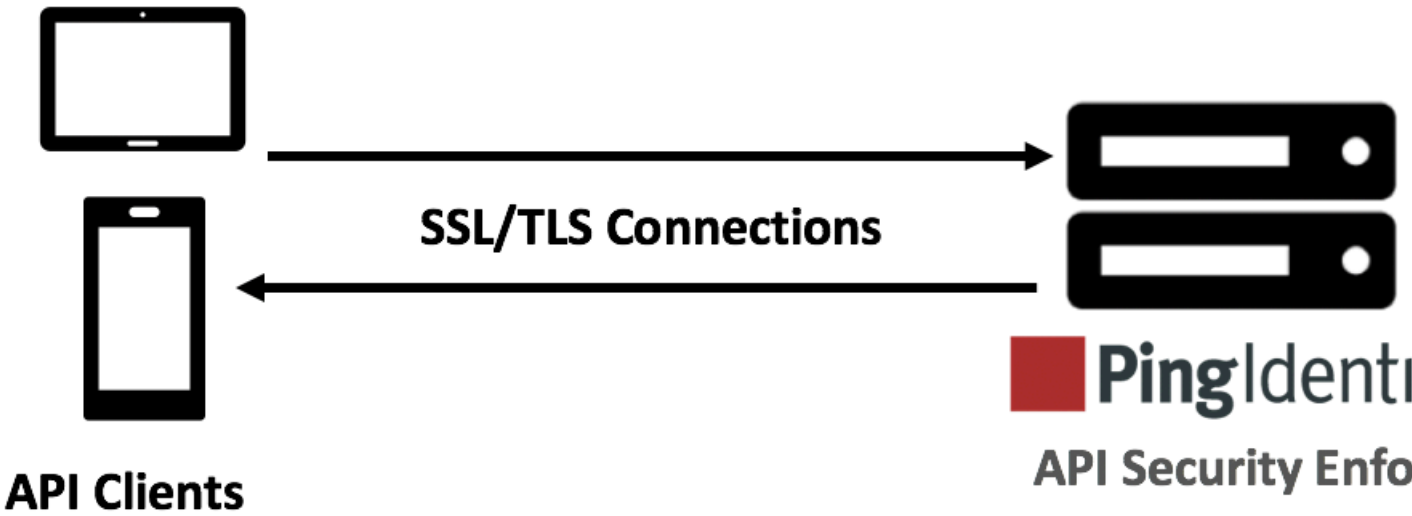
4. Repeat step-2 and step-3 for all the other nodes in the cluster, to complete the cluster restart.

Configure SSL for external APIs

ASE supports both TLS 1.2 and SSLv3 for external APIs. You can configure SSL in ASE for client side connection using one of the following methods:

- **Method 1**: Using CA-signed certificate
- **Method 2**: Using self-signed certificate
- **Method 3**: Importing 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 illustration below:

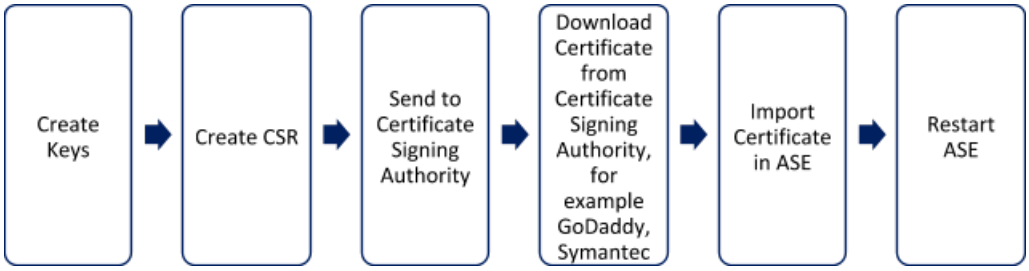


In a cluster setup:

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

Method 1: Use 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:



ⓘ **Note:** ASE internally validates the authenticity of the imported certificate.

To use a CA-signed certificate:

1. Create a private key. ASE CLI is used to create a 2048-bit private key and to store it in the keystore.

```
/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.

```
/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 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 keystore.

```
/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.

Method 2: Use self-signed certificate

A self-signed certificate is also supported for customer testing.

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 `/opt/pingidentity/ase/config/certs/dataplane/dh1024.pem` directory.

```
/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:

```
/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. Create a self-signed certificate. Use the CLI to produce a self-signed certificate using the certificate request 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
```

4. Restart ASE by stopping and starting.

Method 3: Import 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. Import key pair:

```
/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 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
```

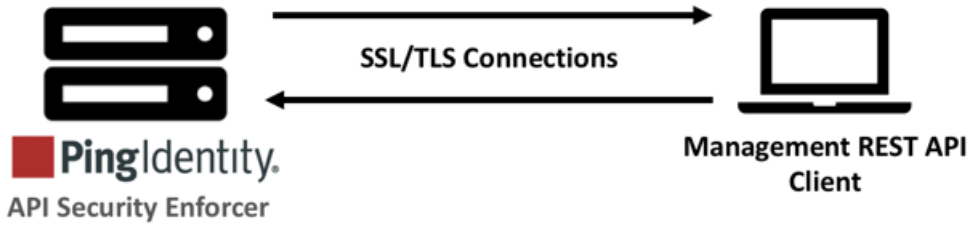
3. Restart ASE by stopping and starting.

Configure SSL for management APIs

ASE supports both TLS 1.2 for management APIs. You can configure SSL in ASE for *management APIs* using one of the following methods:

- **Method 1**: Using CA-signed certificate
- **Method 2**: Using self-signed certificate
- **Method 3**: Importing 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

Method 1: Use 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:

ⓘ **Note:** ASE internally validates the authenticity of the imported certificate.

To use a CA-signed certificate:

1. Create a private key. ASE 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.

Method 2: Use self-signed certificate

A self-signed certificate is also supported for customer testing.

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
password >
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_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/dataplane/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/management.csr`

```
/opt/pingidentity/ase/bin/cli.sh create_management_self_sign_cert -u admin
 -p
password >
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.

Method 3: Import 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
```

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.

Configure native and PAM authentication

ASE provides two types of authentication:

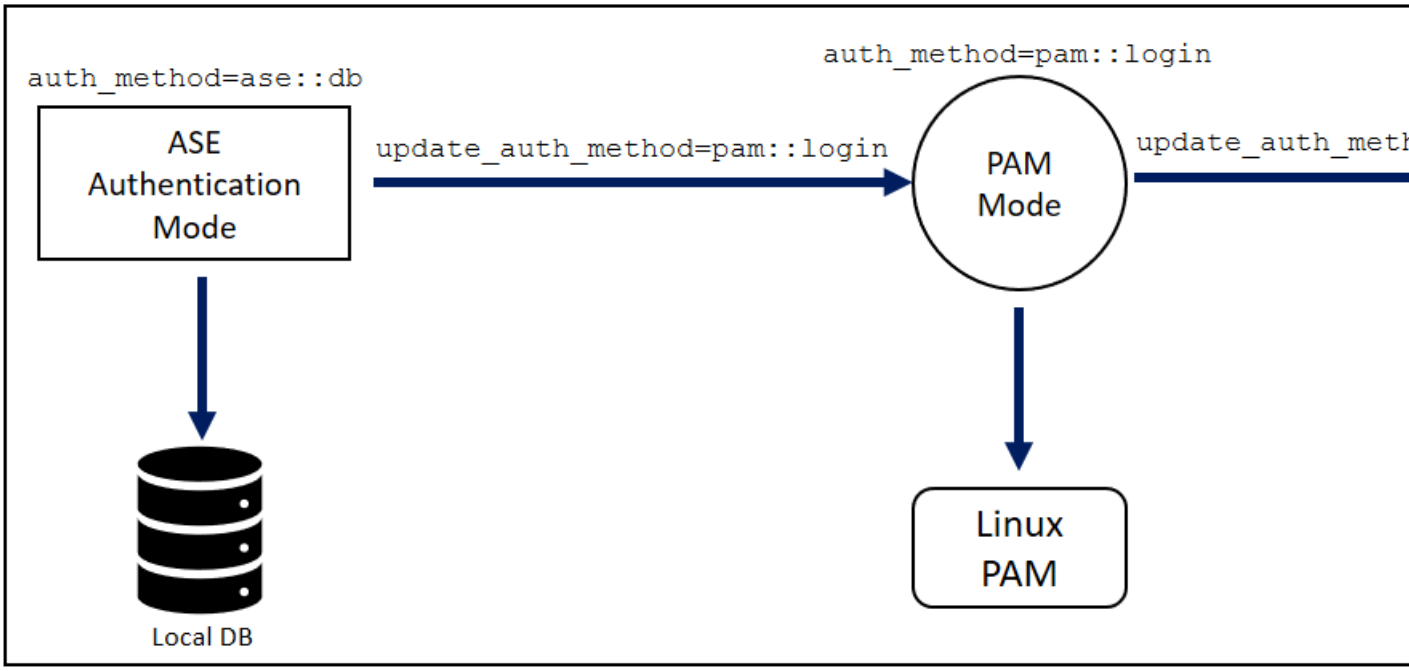
- Linux Pluggable Authentication Module (PAM)
- ASE native authentication (default method)

All actions carried out on ASE require an authenticated user.

The two methods to choose the authentication method include:

- Configure auth_method parameter in `ase.conf` (see ASE Initial Configuration)
- Execute a CLI command (`update_auth_method` *<method>*).

The sections below provide more details on configuring the desired method. The following diagram shows the transition between authentication modes. The authentication method can be changed during run-time without restarting ASE.



ASE native authentication

By default, ASE uses native ASE authentication which ships with the system. Each user can execute CLI commands by including the shared “username” and “password” with each command. The system ships with a default username (`admin`) and password (`admin`). Always change the default password using the **`update_password`** command. For more information on ASE commands, see Appendix A.

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. 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>
```

Here 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>
```

Linux Pluggable Authentication Modules (PAM) authentication

PAM-based authentication provides the flexibility to authenticate administrators using existing authentication servers, such as your organization's 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.

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. Service scripts include `login, su, ldap,` etc. For example, `login` script allows all system users administrative access to ASE. To support PAM authentication with `login` script, update auth_method configuration values in `ase.conf` :

```
auth_method=pam::login
```

Here 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 (for example **`ldap`**) which defines PAM's behavior for user authentication to the `/etc/pam.d` directory. To set the authentication method and use the **`ldap`** script, enter the following command:

```
/opt/pingidentity/ase/bin/cli.sh update_auth_method pam::ldap -u admin -p
password>
```

Here is a snippet of a sample script:

```
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.
```

In the above example, the PAM module uses the organization's LDAP server to authenticate users.

Recovering ASE from unavailable `pam.d`script

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, copy `etc/pam.d/login` to `etc/pam.d/other`. ASE will now use the credentials in `etc/pam.d/login` to authenticate administrators. After logging back into ASE, change the authentication method to

use the correct file name. Copying the contents of `etc/pam.d/login` to `etc/pam.d/other` does not need a restart of ASE or the host operating system.

ASE management, access and audit logs

ASE generates two types of logs:

- **Access log** contains information about all API traffic
- **Management log** contains information about Controller and Balancers

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`. Examples for port 80 and port 443 are:

- `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 `abs_uploaded` directory in the `logs` directory.

The following snippet shows an example log file:

```
-rw-r--r--. 1 root root 0 Aug 10 13:10
 http_ws_80_pid_0__access__2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10
 https_wss_443_pid_0__access__2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10
 http_ws_80_pid_19010__access__2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10
 http_ws_80_pid_19009__access__2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10
 https_wss_443_pid_19022__access__2018-01-22_13-10.log
-rw-r--r--. 1 root root 0 Aug 10 13:10
 https_wss_443_pid_19017__access__2018-01-22_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
```

Management logs

Management log detail levels (for example INFO, WARNING, 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

`controller.log` is a log file with data from the 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

`balancer.log` for port 80 and `balancer_ssl.log` for port 443 are static files which 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, balancer checks for request-response parsing error at every 30-second. Parsing error statistics is logged in `balancer.log` file only if balancer encounters parsing errors. If there are no errors in a 30-second period, the `balancer.log` file does not show the JSON output. 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
    }
  }
}
```

The snippet shows that in-total there was one parsing error for request and one for the response. The statistics also lists the type of request and response error.

Audit logs

ASE logs administrator actions (for example CLI commands, 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
```

Change management log levels

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. The default value is `INFO`:

- FATAL
- ERROR
- WARNING
- INFO
- DEBUG

You can change the log level of management logs during run-time by using the **`log_level`** command. The **`log_level`** command works in an identical way for both sideband and inline ASE modes. In an ASE cluster set up, run the **`log_level`** command on all the ASE nodes. The change in log-level is also recorded in Audit logs. Following is an example CLI output of the **`log_level`** command to change the log-level to `warn`. The other values for the command are `info`, `error`, `fatal`, and `debug`.

```
#./bin/cli.sh -u admin -p admin log_level warn
```

You can also verify the current log level by using the ASE **`status`** command.

```
#./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)
```

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 ABS. 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 ASE command line:

```
/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_27889__2017-04-01_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_27905__access__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.

Configure syslog

Syslog messages are a standard for sending event notification messages. These messages can be stored locally or on an external syslog server. ASE generates and sends syslog messages to an external syslog server over UDP. All the syslog messages sent belong to the informational category.

Configuring syslog server

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. To stop generating syslog messages, remove the syslog server definition from the `ase.conf` file, stop and then start ASE. Here 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=
```

Listing syslog server

Show the configured syslog server by executing the **`list_sys_log_server`** command:

```
/opt/pingidentity/bin/cli.sh list_syslog_server -u admin -p
192.168.11.108:514, messages sent: 4, bytes sent: 565
```

Here 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
```

Email alerts and reports

ASE sends email notifications under two categories:

- **Alerts** – alerts are event based.
- **Reports** – sent at a configured frequency (`email_report`) from one to seven days.

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 *ASE cluster setup* on page 121.

```
; 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

Email alerts are sent based on the following event categories:

- **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 `ase.conf`. The default value is 180 minutes. 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 not able to send access log files to 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 last one hour. The **total** counter lists the total number of times that error has occurred since ASE was started.

- ABS seed node resolve
- ABS authentication
- 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

Following is a template for alerts:

```
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 occured>
```

For example,

```
Event : high memory usage
Value : 82.19%
When : 2019-May-16 18:30:00 PST
Where : vortex-132
```

Alerts logged in log file: Following is a list of all the alerts that are logged in `controller.log` file when email alerts are disabled (`enable_email=false`) in `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

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. 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

Following is a template of weekly or daily email report:

```
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
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 ase.conf file by configuring `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

Following is a sample email template for decoy API:

```
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
```

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 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 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.
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

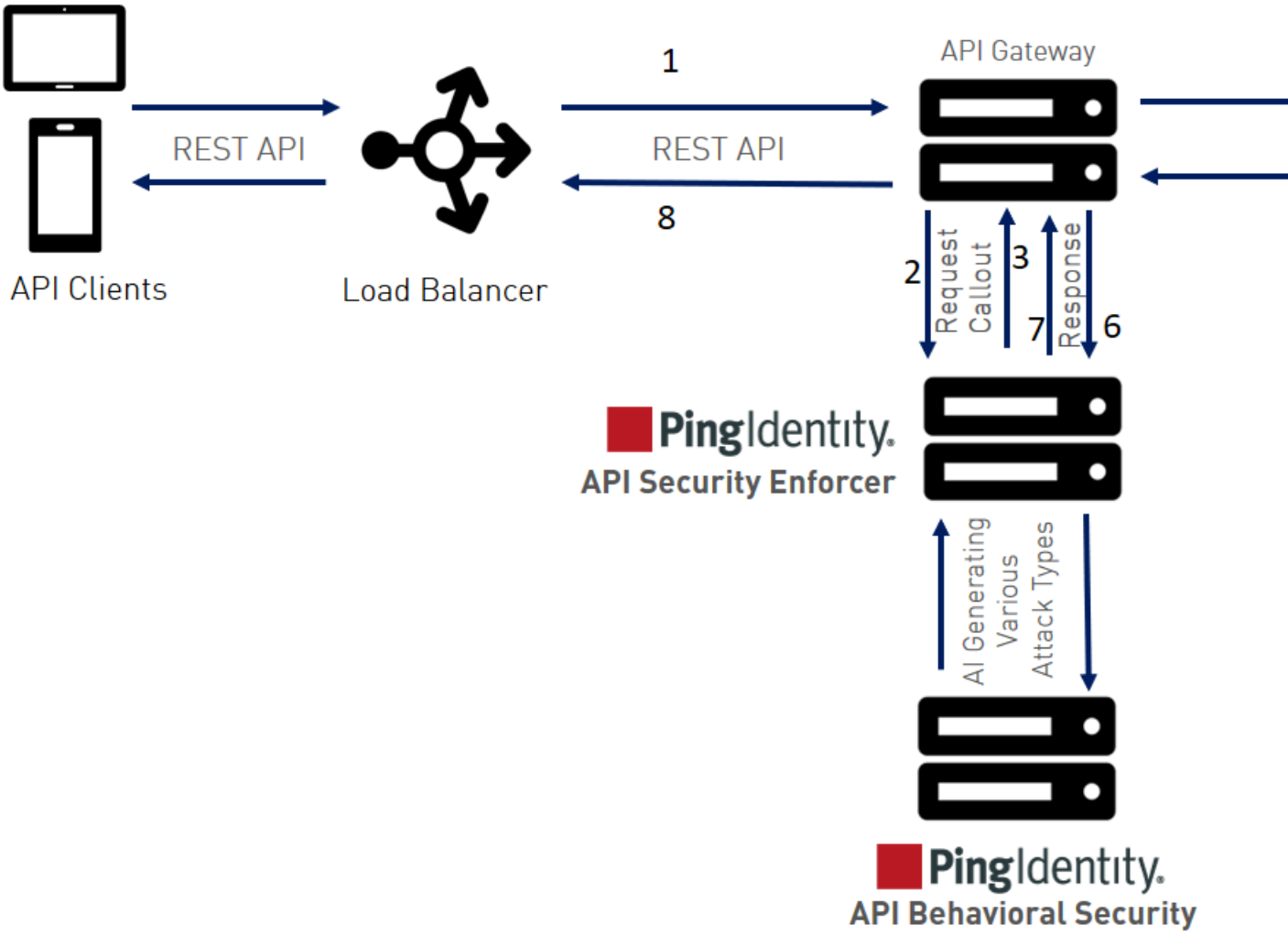
ASE sends one or more alerts when it is not able to send access log files to ABS. The following table lists the alerts and possible resolution for the alerts.

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 hostname provided in `abs.conf` 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.
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 ASE receives 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 shows a typical ASE sideband deployment:



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

1. Incoming request to API gateway
2. 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 IP against the AI generated Blacklist. 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 AI engine for processing.
7. ASE receives the metadata 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:** Make sure that XFF is enabled in the API gateway for ASE to detect the client IP addresses correctly.

Configuring ASE for sideband

To configure ASE to work in the sideband mode, edit the `ase.conf` file located in the `config` directory. Set the value of the mode parameter to `sideband`. The default value of the mode parameter is `inline`. 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
```

Enable sideband authentication

To have a secure the connection between your API gateway and ASE, enable sideband authentication in ASE and generate a sideband token. This token is configured in the API gateway for it to communicate securely with ASE.

```
/opt/pingidentity/ase/bin/cli.sh enable_sideband_authentication -u admin -p
 admin
Sideband authentication is successfully enabled
```

Generate sideband token: Enter the following command to generate ASE sideband token:

```
/opt/pingidentity/ase/bin/cli.sh create_sideband_token -u admin -p admin
Sideband token d9b7203c97844434bd1ef9466829e019 created.
```

ASE configuration - ase.conf

To secure your API environment using sideband ASE deployment, APIs need to be configured in API security Enforcer using an API JSON file. Each API has a unique API JSON file. For example, 5 APIs would require configuration of 5 API JSON files. ASE ships with sample JSON files located in the `/config/api` directory. Two options exist for deploying API JSON files:

1. Automated deployment using AAD which is documented in the ABS Engine admin guide
2. Manually configure the JSON file with the required parameters as shown in the next section.

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 application requirements. The parameter values and descriptions are included in the following table:

Parameter	Description
ASE mode	
mode	Change the mode to `sideband` for ASE to work in a sideband mode. T default value is `inline`.
ASE timezone	
timezone	Sets ASE's timezone. The values can be `local` or `UTC`. Default value i `local`. If ASE is deployed in a cluster, configure the same timezone or 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` enable authentication between in client, for example, an API gateway a ASE. After setting it to `true`, generate a sideband authentication token using ASE **`create_sideband_token`** command.
ASE ports	
http_ws_port	Data port used for http or WebSocket protocol. The default value is 80.
https_wss_port	Data port used for https or Secure WebSocket (wss). The default value is 443.
management_port	Management port used for 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
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. 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)
ase_health	When `true`, enables load balancers to perform a health check using th following URL:"http(s)://<*ASE Name*>/ase" where <*ASE Name*> is the A 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 `tr`

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 keyst password, see *Updating Keystore Password*.
enable_hostname_rewrite	NA
	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`.
server_ca_cert_path	N/A
enable_xff	N/A
enable_firewall	When `true`,activates the ASE firewall. 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 (i.e. 24 hours).
	AI-based API security (ABS)
enable_abs	When `true`, send access log files to ABS for generating API metrics a detecting attacks using machine learning algorithms.
enable_abs_attack	When `true`, ASE fetches attack list from ABS and blocks access by cli 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 defa 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. Th 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`, for example, `/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.
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/Su 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 10,0 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.
	Alerts and reports
enable_email	When `true`, send email notifications. See *Email alerts and reports* on p 137 for more information. The default value is `false`.
email_report	Time interval in days at which ASE sends reports. Minimum value is on 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 su that the SMTP server supports SSL. If you set smtp_ssl to `true` and th SMTP server does not support SSL, email communication falls back to non-SSL channel. The default value is `true`. Set it to false if email communication is over a non-SSL channel. The e communication will fail if you set the parameter to `false`, but the SMTP server only supports SSL communication.
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 certificat however, the communication is still over SSL.

ⓘ **Note:** If you have configured an IP address as `smtp_host` and se `smtp_cert_verification` to `true`, then make sure that the certific 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.

A sample `ase.conf` file is displayed below:

```
; 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=80
https_wss_port=443

; 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)
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=false

; Set this value to true, to allow API Security Enforcer to fetch attack
 list from ABS.
enable_abs_attack=false

; 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 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

; 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:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerMI
```

```
; enable hostname rewrite for inline mode. ASE will rewrite the host header
 in request
; to the server's hostname
enable_hostname_rewrite=false

; Set the timezone to utc or local. The default timezone is local.
timezone=local

; Google Pub Sub Configuation
enable_google_pubsub=false

google_pubsub_topic=/topic/apimetrics

; Number of concurrent connections to Google Pub/Sub
; 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 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”.

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 – API JSON configuration file

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:

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 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 ABS 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 *Configuring OAuth2 Token*.
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 Configuring API keys.
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 *Configuring API keys*.
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_mapping	N/A

API pattern enforcement	N/A
protocol_allowed	
http_redirect	
methods_allowed	
content_type_allowed	
error_code	
error_type	
error_message_body	
Flow control	N/A
client_spike_threshold	
client_connection_queuing	
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	N/A
health_check_interval	
health_retry_count	
health_url	
server_ssl	N/A
Servers:	The IP address or hostname and port number of each backend server running the API.
host	
port	
server_spike_threshold	N/A
server_connection_quota	
Decoy Config	When decoy_enabled is set to `true`, decoy sub-paths function as decoy APIs .
decoy_enabled	
response_code	response_code is the status code (for example `200`) that ASE returns when a decoy API path is accessed.
response_def response_message	
decoy_subpaths	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

JWT location username clientid	When the parameter values of JWTobject are set, ASE decodes the JWT to extract the user information. 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* on page 161.

Here is a sample JSON file for a REST API:

```
{
"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": "",
"response_message": "",
"decoy_subpaths": []
},
"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 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
```

Activate API cybersecurity

API Security Enforcer provides real-time API cybersecurity using the list of attacks generated by PingIntelligence AI engine. Real time API Cyber Security is activated only when ASE firewall is enabled.

Enable API cybersecurity

To enable API security, enter the following command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
Firewall is now enabled
```

After enabling API Security, enter the following CLI command to verify cybersecurity is enabled:

```
/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
```

Disable API cybersecurity

To disable ASE's cybersecurity feature, type the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_firewall
Firewall is now disabled
```

After disabling ASE's cybersecurity feature, enter the following CLI command to verify that cybersecurity is disabled:

```
/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 supports real time ASE attack detection and blocking for API Deception. ASE blocks hackers who probe a decoy API (see *API Deception Environment*) and later try to access a real business API.

Enable ASE detected attacks

Enable real-time ASE attack detection by running the following ASE command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin
 enable_ase_detected_attack
```

ASE detected attack is now enabled

Disable ASE detected attacks

Disable real-time ASE detected attacks by running 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 Blacklist.

Capture client identifiers

ASE identifies attackers for HTTP(s) protocol using five client identifiers:

- Username
- API keys
- OAuth2 token
- Cookie
- IP address

ⓘ **Note:** Username is not configured in the api_metadata object of API JSON. However, ASE supports the extraction of usernames coming in a JSON Web Tokens(JWTs), and a JWT object in API JSON is used to configure username information. For more information, see *Extract user information from JWT in sideband mode* on page 161. For usernames that are not part of the JWTs PingIntelligence 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.

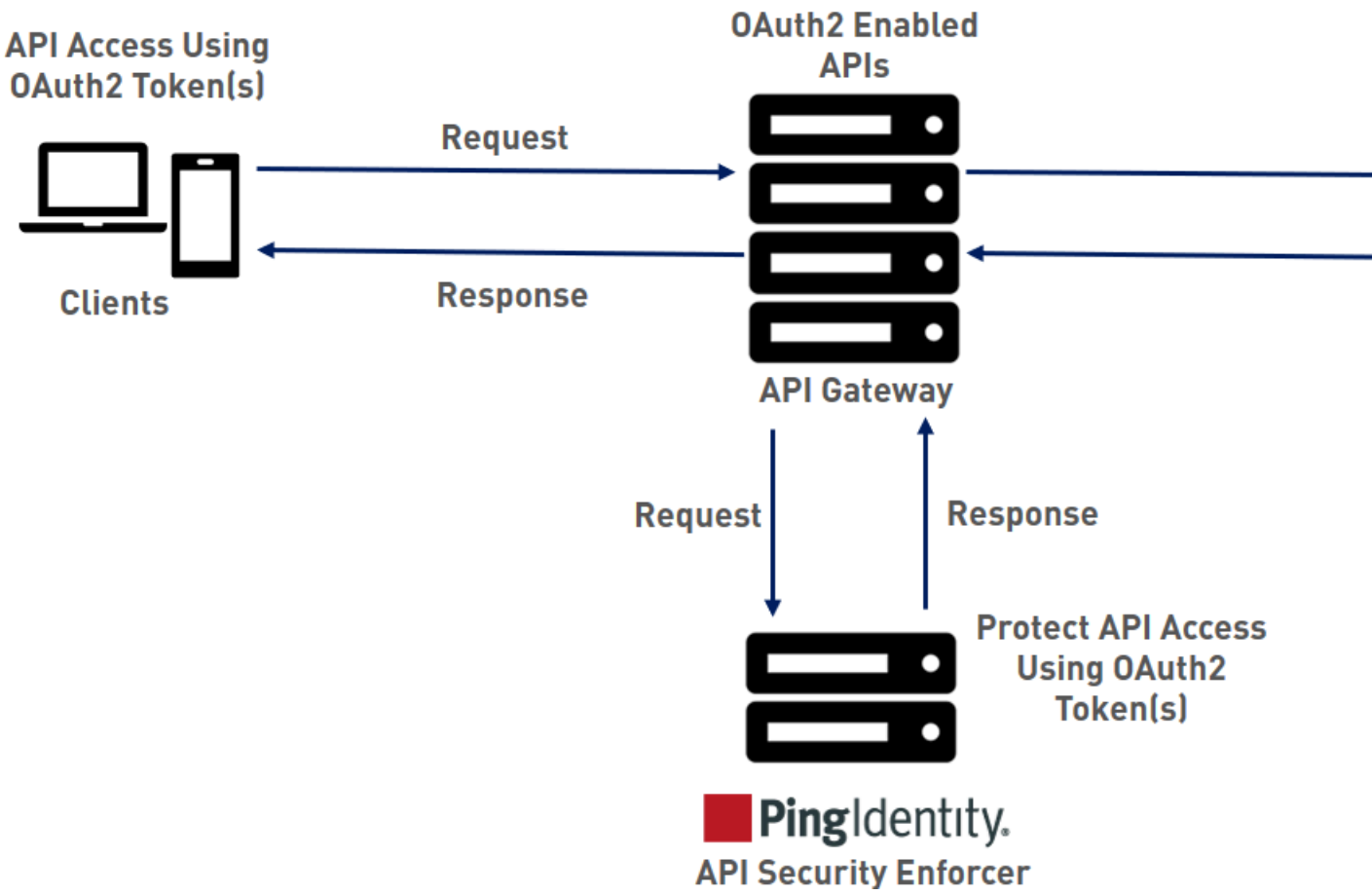
Configure ASE support for OAuth2 tokens

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 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,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
```

When blocking is enabled, ASE checks the token against the list of tokens in the whitelist and blacklist. If the token is in the blacklist, the client using the token is immediately blocked.

The following diagram shows the traffic flow in an OAuth2 environment:



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`).

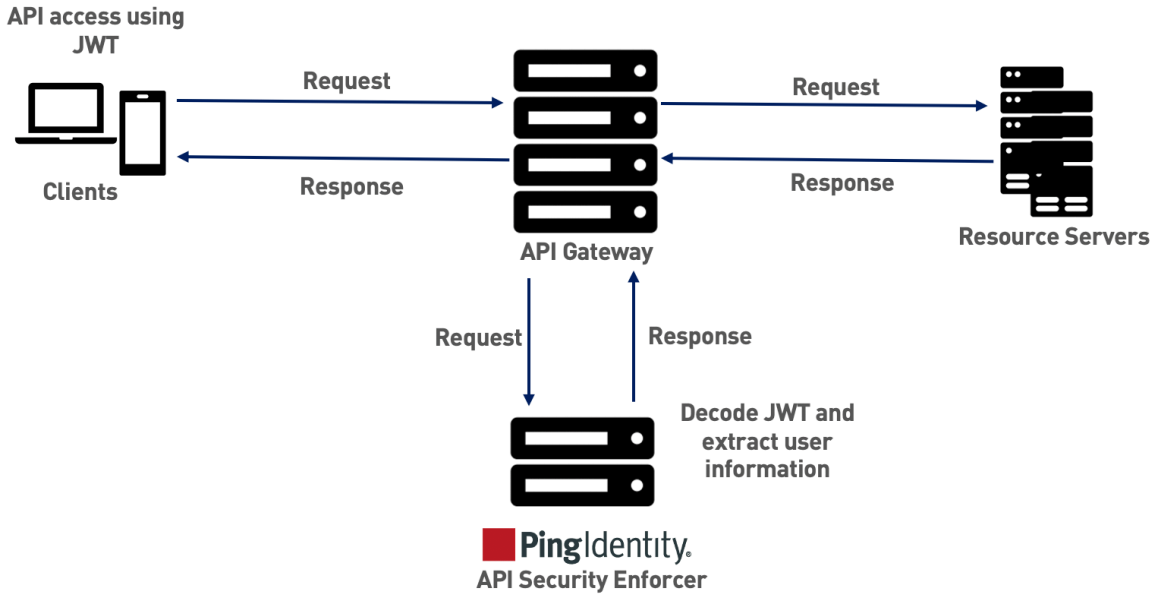
```
"api_metadata": {
 "protocol": "http",
 "url": "/",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": true,
 "oauth2_access_token": true,
 "apikey_qs": "key_1.4",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": 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 blacklist.

Extract user information from JWT in sideband mode

ASE supports the decoding of transparent JSON Web Tokens (JWTs) received as part of API requests. It extracts the user information from the JWT and logs it in ASE access logs. The ABS (API Behavioral Security) AI engine analyses these access logs to generate reports and detect attacks.

The following diagram shows the traffic flow when ASE is in sideband mode.



A JWT consists of three parts - header, payload, and signature. They are concatenated with periods(**.**). The following is 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 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.

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 or 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.
- 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.

Configure API JSON

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 – API JSON configuration file* on page 153.

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": ""
    },
    "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.

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.mFrs3Zodq`

username	It is the JWT claim to extract the username.
clientid	It is 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 whitelist and blacklist. If the username is in the blacklist, the client using the username is blocked.

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. For more information on API discovery, see *API discovery and configuration* on page 321. 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 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": "" }`
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.

Manage whitelist and blacklist

ASE maintains the following two types of lists:

- **Whitelist** – 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 CLI commands.
- **Blacklist** – 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:
  - ABS detected attacks (for example data exfiltration). ABS detected attacks have a time-to-live (TTL) in minutes. The TTL is configured in ABS.
  - ASE detected attacks (for example invalid method, decoy API accessed). The ASE detected attacks
  - List of “bad” clients manually generated by CLI

Manage whitelists

Valid operations for OAuth2 Tokens, cookies, IP addresses, API keys, and usernames on a whitelist include:

Add an entry

- Add an IP address to whitelist:

```
/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 whitelist:

```
/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
```

- Add a token to whitelist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist token
 token1.4
token token1.4 added to whitelist
```

- Add an API Key to whitelist:

```
/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
```

- Add a username to whitelist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist username
 abc@example.com
username abc@example.com added to whitelist
```

View whitelist

```
/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
```

Delete an entry

```
/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
```

Clear the whitelist

```
/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
```

Manage blacklists

Valid operations for IP addresses, Cookies, OAuth2 Tokens, and API keys on a blacklist include:

Add an entry

- Add an IP address to blacklist:

```
/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
```

- Add a cookie to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist cookie
 JSESSIONID ad233edqsd1d23redwefew
cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
```

- Add a token to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token
 ad233edqsd1d23redwefew
token ad233edqsd1d23redwefew added to blacklist
```

- Add an API Key to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist api_key
 AccessKey b31dfa4678b24aa5a2daa06aba1857d4
api_key AccessKey b31dfa4678b24aa5a2daa06aba1857d4 added to blacklist
```

- Add an username to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist username
 abc@example.com
username abc@example.com added to blacklist
```

ⓘ **Note:** You can also add username with space to blacklist. For example, "your name".

View blacklist - entire blacklist or based on the type of real time violation.

```
/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
```

Blacklist based on decoy IP addresses

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist decoy
Realtime Decoy Blacklist
1) type : ip, value : 4.4.4.4
```

Blacklist based on protocol violations

```
/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
```

Blacklist based on method violations

```
/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
```

Blacklist based on content-type violation

```
/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
```

ABS detected attacks

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist
 abs_detected
No Blacklist
```

Delete an entry

```
/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
```

Clear the blacklist

```
./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
```

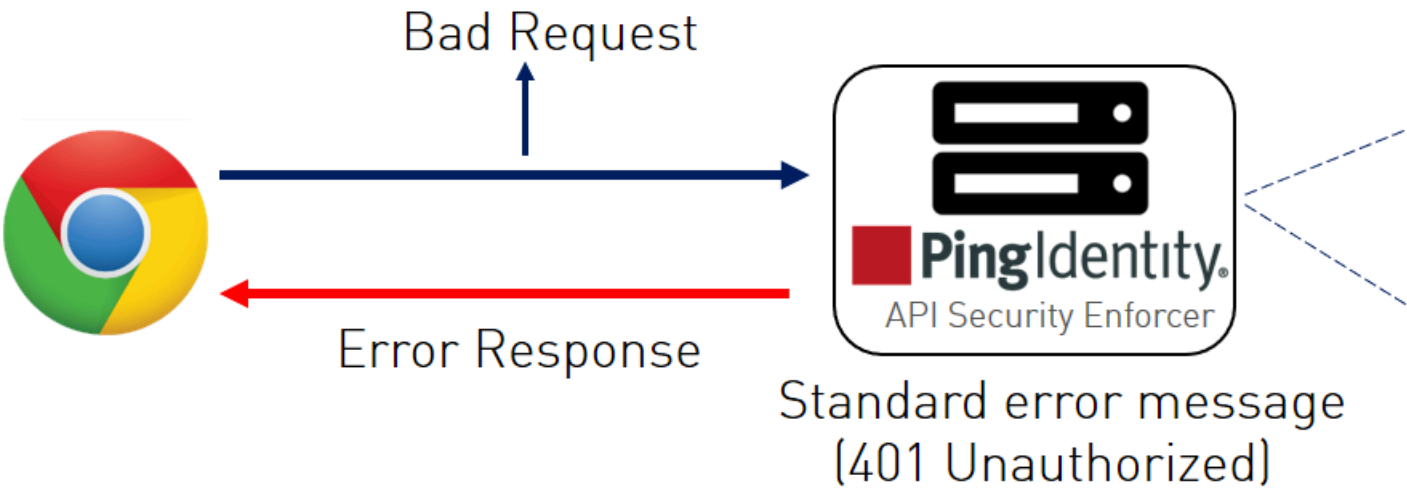
When clearing the blacklist, make sure that the real-time ASE detected attacks and ABS detected attacks are disabled. If not disabled, the blacklist gets populated again as both ASE and ABS are continuously detecting attacks.

ASE generated error messages for blocked requests

ASE blocks certain requests based on API Mapping or 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 Hostn
Malformed Request	400	Bad Request	Error: Malformed Req
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



Per API blocking

ASE can be configured to selectively block on a per API basis by configuring an API JSON file parameter. To enable per API blocking for each API, set the enable_blocking parameter to `true` in the API JSON. For 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": ""
 },
```

If per API blocking is disabled, ABS still detect attacks for that specific API, however, ASE does not block them. ASE will continue to block attacks on other APIs with the enable_blocking set to `true`.

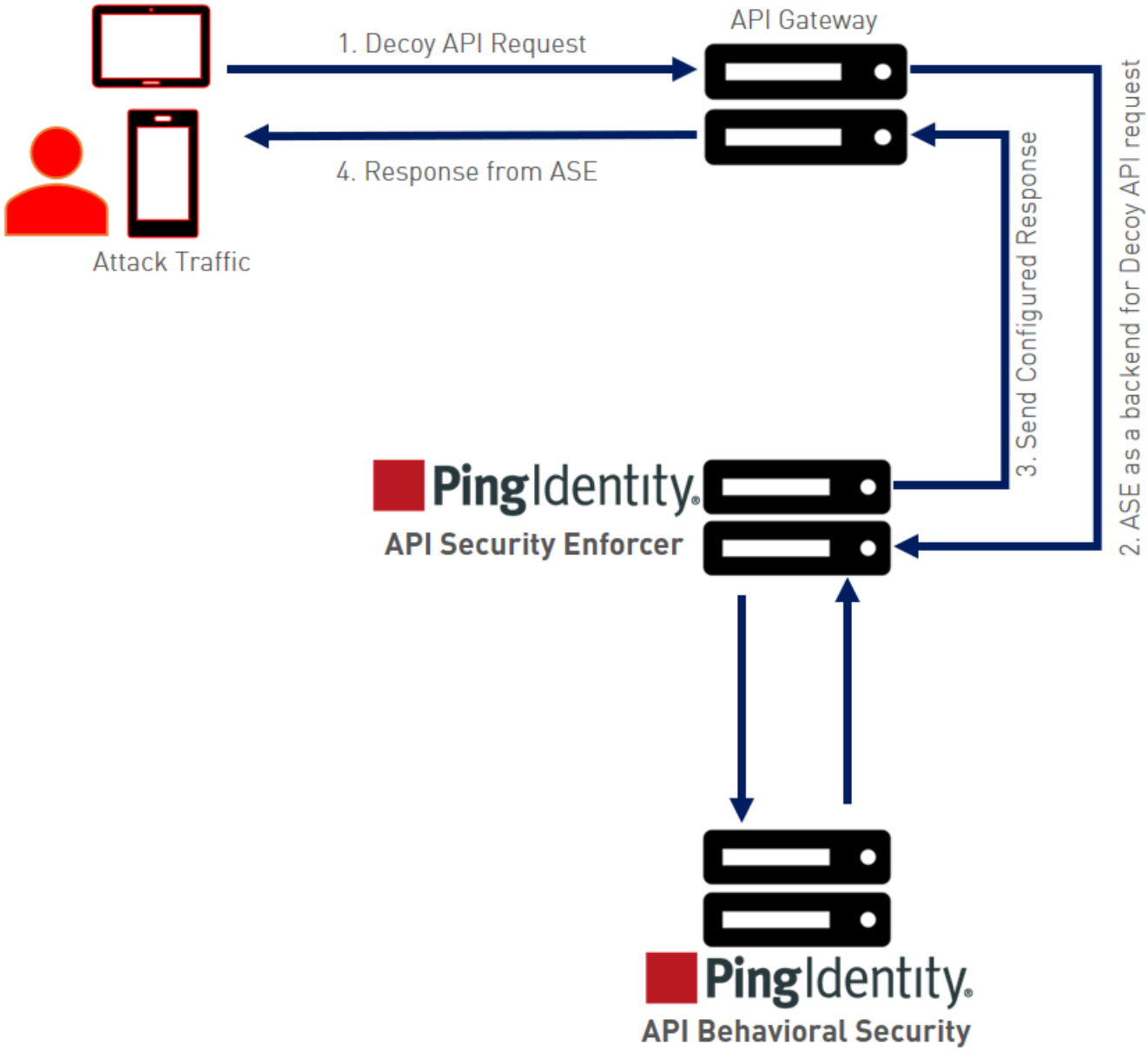
API deception environment

A decoy API is configured in 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_messageparameter in API JSON 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_8787__2017-04-04_10-57.log)`. Decoy log files are rotated every 24-hours and stored in the `opt/pingidentity/ase/logs` directory.

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.

ⓘ **Note:** In sideband ASE deployment you can configure only out-of-context decoy API.



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 ABS for further analysis. Following is a snippet of an API JSON file which has been deployed as an out-of-context 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 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
```

Real-time API deception attack blocking

When a client probes a decoy API, ASE logs but does not drop the client connection. However, if the same client tries to access a legitimate business API, then ASE block the client in real-time. Here is a snippet of an ASE access log file showing real time decoy blocking:

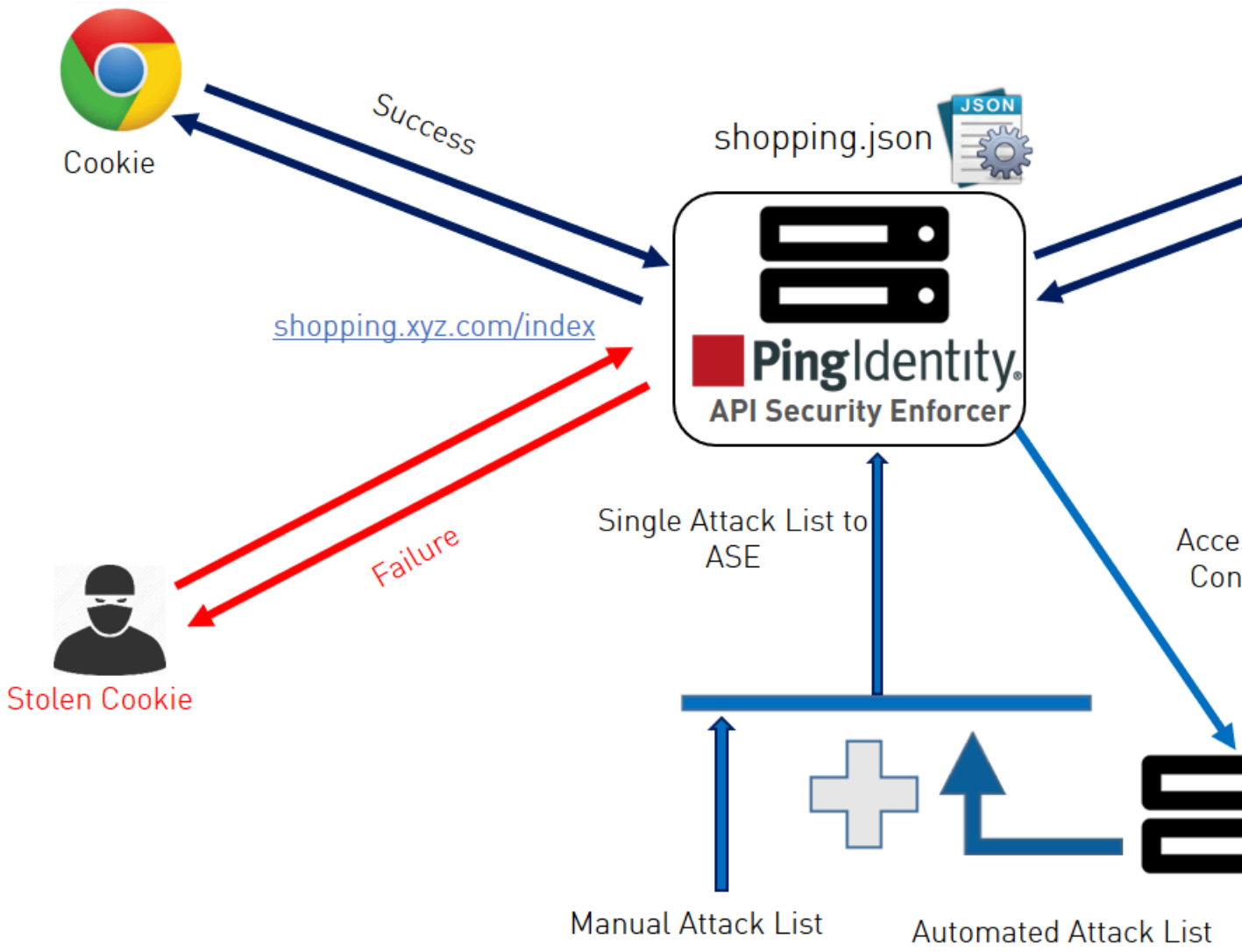
```
[Tue Aug 1422:51:49:707 2018] [thread:209] [info] [connectionid:1804289383]
 [connectinfo:100.100.1.1:36663] [type:connection_drop] [api:decoy]
 [request_payload_length:0] GET /decoy/test/test HTTP/1.1
User-Agent: curl/7.35.0
Accept: */*
Host: app
```

The blocked client is added to the blacklist 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

ABS AI engine detects attacks using artificial intelligence (AI) algorithms. After receiving ASE access logs and API JSON configuration files, ABS applies AI algorithms to track API connections and detect attacks. If enable_abs_attack is `true`, ABS sends blacklist to ASE which blocks client identifiers, like, API keys, usernames, cookie, IP address, and OAuth token on the list.



Configure ASE to ABS connectivity

To connect ASE to ABS, configure the ABS address (IPv4:Port or Hostname:Port), access key, and secret key in the `abs.conf` file located in the `/opt/pingidentity/ase/config` directory.

ⓘ **Note:** enable_absmust be set to `true` in the `ase.conf` file. when ABS is in a different AWS security group, use a private IP address

The parameter values and descriptions are included in the following table:

Parameter	Description
abs_endpoint	Hostname and port or the IPv4 and port of all the ABS nodes

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 ⓘ **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 the ABS nodes. The same value has to be configured in ABS Mongo database. This value is obfuscated during the start of ASE. ⓘ **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 the value is set to false.
abs_ca_cert_path	Location of the trusted 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.

ⓘ **Note:** The access_key and secret_key are configured in ABS. For more information, see *ABS Admin Guide*.

Here 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.
; 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-ABS encrypted communication

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`. The abs_ca_cert_path is the location of ABS's trusted CA certificate. If the field is left empty, ASE does not verify ABS's certificate, however, the communication is till encrypted.

Check and open ABS ports

The default ports for connection with ABS are 8080 and 9090. Run the **`check_ports_ase.sh`** script on the ASE machine to determine ABS accessibility. Input ABS host IP address and ports as arguments.

```
/opt/pingidentity/ase/util ./check_ports_ase.sh {ABS IPv4:[port]}
```

Manage ASE blocking of ABS detected attacks

To configure ASE to automatically fetch and block ABS detected attacks, complete the following steps:

1. Enable ASE Security. Enter the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
   ```

2. Enable ASE to send API traffic information to ABS. Enter the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs
   ```

3. Enable ASE to fetch and block ABS detected attacks. Enter the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs_attack
   ```

After enabling automated attack blocking, ASE periodically fetches the attack list from ABS and blocks the identified connections. 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.

```
; This value determines how often ASE will query ABS.
abs_attack_request_minutes=10
```

Disable attack list fetching from ABS

To disable ASE from fetching the ABS attack list, entering the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs_attack
```

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. 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 Admin Guide.*

To stop an ASE cluster from sending log files to ABS, enter the following ASE CLI command.

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs
```

After entering this command, ABS will not receive any logs from ASE. Refer to the ABS documentation for information on types of attacks.

Configure 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 `ase.conf` file, ASE sends the event message in a JSON file to Google cloud. You can verify that Google Pub/Sub is enabled by running the ASE `status` command:

```
/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
```

Complete the following steps to configure Google Pub/Sub in ASE:

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. 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 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.

Configure API Key - Optional

You can optionally configure API Key in `ase.conf` file. Obtain the API Key for your Google project and configure in `google_pubsub_apikey` option. Obfuscate the API Key for it to take effect. For more information on obfuscating keys and password, see *Obfuscate keys and passwords* on page 119. Following is a summary of steps that you need to complete:

1. Stop ASE
2. Edit `ase.conf` file to add API Key
3. Obfuscate the API Key
4. Start ASE

ASE JSON message file

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 3-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 management, access and audit logs* on page 134. When messages are successfully published to Google Pub/Sub, the message ID is logged in success log file. Following is a snippet of event message JSON file logged in `balancer.log` file when ASE is run in debug mode.

```
{
  "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"
}
```

CLI for sideband ASE

Start ASE

Description

Start ASE

Syntax

```
./start.sh
```

Stop ASE

Description

Stop ASE

Syntax

```
./stop.sh
```

Help

Description

Displays cli.sh help

Syntax

```
./cli.sh help
```

Version

Description

Displays the version number of ASE

Syntax

```
./cli.sh version
```

Status

Description

Displays the running status of ASE

Syntax

```
./cli.sh status
```

Update Password

Description

Change ASE admin password

Syntax

```
./cli.sh update_password -u admin - p
```

Change log level

Description

Change balancer.log and controller.log log level

Syntax

```
./cli.sh log_level -u admin -p
```

options - `warn`, `info`, `error`, `fatal`, `debug`

Get Authentication Method

Description

Display the current authentication method

Syntax

```
./cli.sh get_auth_method -u admin -p
```

Update Authentication Method

Description

Update ASE authentication method

Syntax

```
./cli.sh update_auth_method {method} -u admin -p
```

Enable Sideband Authentication

Description

Enable authentication between API gateway and ASE when ASE is deployed in sideband mode

Syntax

```
./cli.sh enable_sideband_authentication -u admin – p
```

Disable Sideband Authentication

Description

Disable authentication between API gateway and ASE when ASE is deployed in sideband mode

Syntax

```
./cli.sh disable_sideband_authentication -u admin – p
```

Create ASE Authentication Token

Description

Create the ASE token that is used to authenticate between the API gateway and ASE

Syntax

```
./cli.sh create_sideband_token -u admin – p
```

List ASE Authentication Token

Description

List the ASE token that is used to authenticate between the API gateway and ASE

Syntax

```
./cli.sh list_sideband_token -u admin – p
```

Delete ASE Authentication Token

Description

Delete the ASE token that is used to authenticate between the API gateway and ASE

Syntax

```
./cli.sh delete_sideband_token {token} -u admin – p
```

Enable Audit Logging

Description

Enable audit logging

Syntax

```
./cli.sh enable_audit -u admin -p admin
```

Disable Audit Logging

Description

Disable audit logging

Syntax

```
./cli.sh disable_audit -u admin -p admin
```

Add Syslog Server

Description

Add a new syslog server

Syntax

```
./cli.sh –u admin -p admin add_syslog_server host:port
```

Delete Syslog Server

Description

Delete the syslog server

Syntax

```
./cli.sh –u admin -p admin delete_syslog_server host:port
```

List Syslog Server

Description

List the current syslog server

Syntax

```
./cli.sh –u admin –p admin list_syslog_server
```

Add API

Description

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

Syntax

```
./cli.sh –u admin –p admin add_api {config_file_path}
```

Update API

Description

Update an API after the API JSON file has been edited and saved

Syntax

```
./cli.sh –u admin –p admin update_api {api_name}
```

List APIs

Description

Lists all APIs configured in ASE

Syntax

```
./cli.sh –u admin –p admin list_api
```

API Info

Description

Displays the API JSON file

Syntax

```
./cli.sh –u admin –p admin api_info {api_id}
```

API Count

Description

Displays the total number of APIs configured

Syntax

```
./cli.sh –u admin –p admin api_count
```

Enable Per API Blocking

Description

Enables attack blocking for the API

Syntax

```
./cli.sh –u admin –p admin enable_blocking {api_id}
```

Disable Per API Blocking

Description

Disable attack blocking for the API

Syntax

```
./cli.sh –u admin –p admin disable_blocking {api_id}
```

Delete API

Description

Delete an API from ASE. Deleting an API removes the corresponding JSON file and deletes all the cookies associated with that API

Syntax

```
./cli.sh –u admin –p admin delete_api {api_id}
```

Generate Master Key

Description

Generate the master obfuscation key `ase_master.key`

Syntax

```
./cli.sh –u admin –p admin generate_obfkey
```

Obfuscate Keys and Password

Description

Obfuscate the keys and passwords configured in various configuration files

Syntax

```
./cli.sh –u admin –p admin obfuscate_keys
```

Create a Key Pair

Description

Creates private key and public key pair in keystore

Syntax

```
./cli.sh –u admin –p admin create_key_pair
```

Create a CSR

Description

Creates a certificate signing request

Syntax

```
./cli.sh –u admin –p admin create_csr
```

Create a Self-Signed Certificate

Description

Creates a self-signed certificate

Syntax

```
./cli.sh –u admin –p admin create_self_sign_cert
```

Import Certificate

Description

Import CA signed certificate into keystore

Syntax

```
./cli.sh –u admin –p admin import_cert {cert_path}
```

Create Management Key Pair

Description

Create a private key for management server

Syntax

```
/cli.sh –u admin –p admin create_management_key_pair
```

Create Management CSR

Description

Create a certificate signing request for management server

Syntax

```
/cli.sh –u admin -p admin create_management_csr
```

Create Management Self-signed Certificate

Description

Create a self-signed certificate for management server

Syntax

```
/cli.sh –u admin -p admin create_management_self_sign_cert
```

Import Management Key Pair

Description

Import a key-pair for management server

Syntax

```
/cli.sh –u admin -p admin import_management_key_pair {key_path}
```

Import Management Certificate

Description

Import CA signed certificate for management server

Syntax

```
/cli.sh –u admin -p admin import_management_cert {cert_path}
```

Cluster Info

Description

Displays information about an ASE cluster

Syntax

```
./cli.sh –u admin -p admin cluster_info
```

Delete Cluster Node

Description

Delete and inactive ASE cluster node

Syntax

```
./cli.sh –u admin -p admin delete_cluster_node host:port
```

Enable Firewall

Description

Enable API firewall. Activates pattern enforcement, API name mapping, manual attack type

Syntax

```
./cli.sh –u admin -p admin enable_firewall
```

Disable Firewall

Description

Disable API firewall

Syntax

```
./cli.sh –u admin -p admin disable_firewall
```

Enable ASE detected attacks

Description

Enable ASE detected attacks

Syntax

```
./cli.sh –u admin -p admin enable_ase_detected_attack
```

Disable ASE Detected Attacks

Description

Disable API firewall

Syntax

```
./cli.sh –u admin -p admin disable_ase_detected_attack
```

Enable ABS

Description

Enable ABS to send access logs to ABS

Syntax

```
./cli.sh –u admin -p admin enable_abs
```

Disable ABS

Description

Disable ABS to stop sending access logs to ABS

Syntax

```
./cli.sh –u admin -p admin disable_abs
```

Adding Blacklist

Description

Add an entry to ASE blacklist using CLI. Valid type values are: IP, Cookie, OAuth2 token, API Key, and username

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

Syntax

```
./cli.sh –u admin -p admin add_blacklist {type}{name}{value}
```

Example

```
/cli.sh –u admin -p admin add_blacklist ip 1.1.1.1
```

Delete Blacklist Entry

Description

Delete entry from the blacklist.

Syntax

```
./cli.sh –u admin -p admin delete_blacklist {type}{name}{value}
```

Example

```
cli.sh –u admin -p delete_blacklist token
 58fcb0cb97c54afbb88c07a4f2d73c35
```

Clear Blacklist

Description

Clear all the entries from the blacklist

Syntax

```
./cli.sh –u admin -p admin clear_blacklist
```

View Blacklist

Description

View the entire blacklist or view a blacklist for the specified attack type (for example, invalid_method)

Syntax

```
./cli.sh –u admin -p admin view_blacklist {all|manual|abs_generated|
invalid_content_type|invalid_method|invalid_protocol|decoy}
```

Adding Whitelist

Description

Add an entry to ASE whitelist using CLI. Valid type values are: IP, cookie, OAuth2 token, API key, and username

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

Syntax

```
./cli.sh –u admin -p admin add_whitelist {type}{name}{value}
```

Example

```
/cli.sh -u admin -p admin add_whitelist api_key AccessKey
 065f73cdf39e486f9d7cda97d2dd1597
```

Delete Whitelist Entry

Description

Delete entry from the whitelist

Syntax

```
./cli.sh –u admin -p admin delete_whitelist {type}{name}{value}
```

Example

```
/cli.sh -u admin -p delete_whitelist token
 58fcb0cb97c54afbb88c07a4f2d73c35
```

Clear Whitelist

Description

Clear all the entries from the whitelist

Syntax

```
./cli.sh –u admin -p admin clear_whitelist
```

View Whitelist

Description

View the entire whitelist

Syntax

```
./cli.sh –u admin -p admin view_whitelist
```

ABS Info

Description

Displays ABS status information.

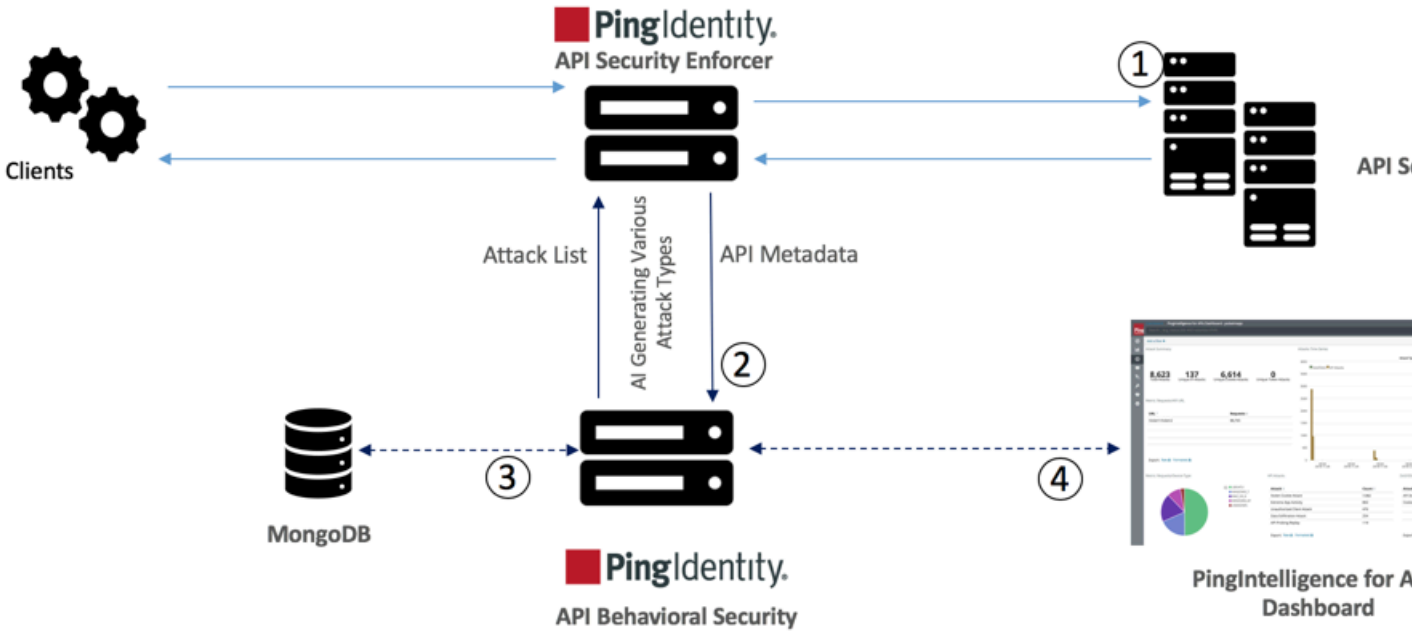
ABS enabled or disabled, ASE fetching ABS attack types, and ABS cluster information

Syntax

```
./cli.sh -u admin -p admin abs_info
```

Inline ASE

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 balancers such as AWS ELB. ASE deployed at the edge of the datacenter, terminates SSL connections from API clients. It then forwards routes the requests directly to the correct destination APIs – and app servers such as Node.js, WebLogic, Tomcat, PHP, etc.



API Security Enforcer
Inline Deployment Mode

To configure ASE to work in the Inline mode, set the mode=inline in the `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 `ase_health` variable in the `ase.conf` file. When `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.

ASE configuration - ase.conf

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 parameter values and descriptions are included in the following table:

Parameter	Description
ASE mode	
mode	The mode in which ASE works. Possible values are `inline` and `sideband`. The default value is `inline`.

ASE timezone	
timezone	Sets ASE's timezone. The values can be `local` or `UTC`. Default value `local`. If ASE is deployed in a cluster, configure the same timezone each cluster node manually.
enable_sideband_keepalive	NA
enable_sideband_authentication	NA
ASE ports	
http_ws_port	Data port used for http or WebSocket protocol. The default value is 80.
https_wss_port	Data port used for https or Secure WebSocket (wss). The default value is 443.
management_port	Management port used for 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
enable_audit	When set to true, ASE logs all actions performed in ASE in the audit l 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 syslo
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 option ▪ 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 following URL: http(s)://*<ASE Name>*/ase where *<ASE Name>* is the domain name The default value is `false`. ⓘ **Note:** Do not configure the /ase URL in an API JSON file.

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. Defau 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 `t`
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 keys password, see *Updating Keystore Password*.

enable_hostname_rewrite	When set to `true`, ASE rewrites the host header in the client request the IP or host and port number configured in the `server` section of th 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 stan 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 por 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 w example.com:8080. **server_ssl** in API JSON set to `true`: ▪ In the server section of API JSON, if the configured port is the stan 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 por 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 w 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 SSLv3. Default value is `false`.
server_ca_cert_path	Location of the trusted CA certificates for SSL/TLS connections from A 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 back server.
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`

	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 pro 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 security (ABS)	
enable_abs	When `true`, send access log files to ABS for generating API metrics a detecting attacks using machine learning algorithms.
enable_abs_attack	When `true`, ASE fetches attack list from ABS and blocks access by t 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 de value is 10-minutes.
Google Pub/Sub configuration	
enable_google_pubsub	NA
google_pubsub_topic	NA
google_pubsub_concurrency	NA
google_pubsub_qps	NA
google_pubsub_apikey	NA
cache_queue_size	NA
google_pubsub_timeout	NA
	Alerts and reports
enable_email	When `true`, send email notifications. The default value is `false`. AS logs the alerts in `balancer.log` file even when email alerts are disa See *Email alerts and reports* on page 137 for more information.
email_report	Time interval in days at which ASE sends reports. Minimum value is 1 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 s that the SMTP server supports SSL. If you set smtp_ssl to `true` and SMTP server does not support SSL, email communication falls back t non-SSL channel. The default value is `true`. Set it to false if email communication is over a non-SSL channel. The communication will fail if you set the parameter to `false`, but the SM server only supports SSL communication.

smtp_cert_verification	Set to `true` if you want ASE to verify the SMTP server's SSL certifica
	The default value is `true`.
	If you set it to `false`, ASE does not verify SMTP server's SSL certific
	however, the communication is still over SSL.
	ⓘ **Note:** If you have configured an IP address as `smtp_host` and s
	`smtp_cert_verification` to `true`, then make sure that the certifi
	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 no 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:

```
; 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=80
https_wss_port=443

; 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)
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=false

; Set this value to true, to allow API Security Enforcer to fetch attack
 list from ABS.
enable_abs_attack=false

; 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 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

; 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:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerMI

; enable hostname rewrite for inline mode. ASE will rewrite the host header
 in request
; to the server's hostname
enable_hostname_rewrite=false

; Set the timezone to utc or local. The default timezone is local.
timezone=local

; Google Pub Sub Configuation
enable_google_pubsub=false

google_pubsub_topic=/topic/apimetrics

; Number of concurrent connections to Google Pub/Sub
; 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 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”.

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

Define an Inline API JSON configuration file

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:

Parameter	Description
protocol	API request type with `ws` - WebSocket ; `htt`
url	The value of the URL `"/shopping"`- nar `"/shopping/elect` `"/"` – entire server (
hostname	Hostname for the API “`*`” matches any hos
cookie	Name of cookie used
cookie_idle_timeout	The amount of time a The time duration forr s: seconds, m: minute ▪ w: week ▪ mnt: month ▪ yr: year
logout_api_enabled	When `true`, ASE exp
cookie_persistence_enabled	When `true`, the sub
oauth2_access_token	When `true`, ASE cap When `false`, ASE do For more information,
apikey_qs	When API Key is sen For more information,
apikey_header	When API Key is part For more information,
login_url	Public URL used by a

enable_blocking	When `true`, ASE blo Default value is false.
api_memory_size	Maximum ASE memo The default value is 1
health_check	When `true`, enable When `false`, no hea Ping Identity recomm
health_check_interval	The interval in second
health_retry_count	The number of times
health_url	The URL used by AS
health_check_headers	Configure one or mor only to inline ASE dep
server_ssl	When set to true, ASE server.

```
"health_check_h
            "X-
            "X-
        },
```

Example

Example Key
X-Host
X-Custom-Header

Servers:	The IP address or hos
host	See *REST API Protec*
port	
server_spike_threshold	
server_connection_quota	
API Mapping:	Internal URL is mapp
internal_url	See *API Name Mappi*
The following API Pattern Enforcement parameters only apply when API Firewall is activated	
Flow Control	ASE flow control ensu
client_spike_threshold	See *WebSocket API*
server_connection_queueing	
bytes_in_threshold	
bytes_out_threshold	
protocol_allowed	List of accepted proto Values can be HTTP, ⓘ **Note:** When Fire
http_redirect	Redirect unencrypted
response_code	See *Configuring Patte*
response_def	
https_url	
methods_allowed	List of accepted RES `GET, POST, PUT,`
content_type_allowed	List of content types a
error_code	Error message gener
error_type	See *ASE Detected Er*
error_message_body	
Decoy Config	When decoy_enabled
decoy_enabled	response_code is the
response_code	response_def is the r
response_def response_message	response_message is
decoy_subpaths	decoy_subpaths is th See *Configuring API*

JWT	When the parameter
location	location is the place c
username	▪ qs:<key name> ▪ h:<custom hea ▪ h:authorizati ▪ h:authorizati ▪ h:cookie:<coo
clientid	username is the JWT clientid is the JWT cla For more information,

Here is a sample JSON file for a REST API:

```
{
  "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": "",
      "response_message": "",
      "decoy_subpaths": []
    },
    "jwt": {
      "location": "h:authorization:bearer",
      "username": "username",
      "clientid": "client_id"
    }
  }
}
```

Add configured 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 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

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
```

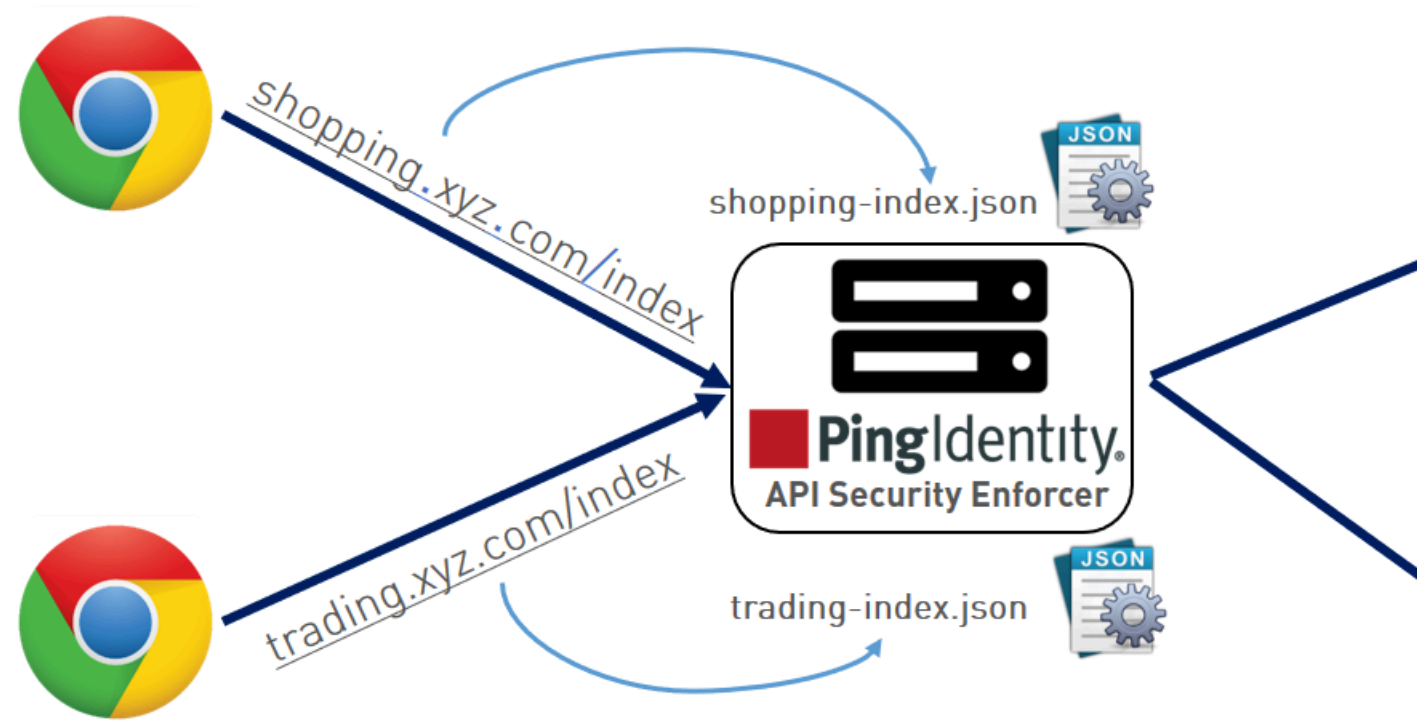
API routing

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 API name.

- *Multiple host names with same API name* for example, shopping.xyz.com/index, trading.xyz.com/index
- *Single host name with different API names* for example, shopping.xyz.com/index, shopping.xyz.com/auth
- *Wildcard host name and 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 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.



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. 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,
```

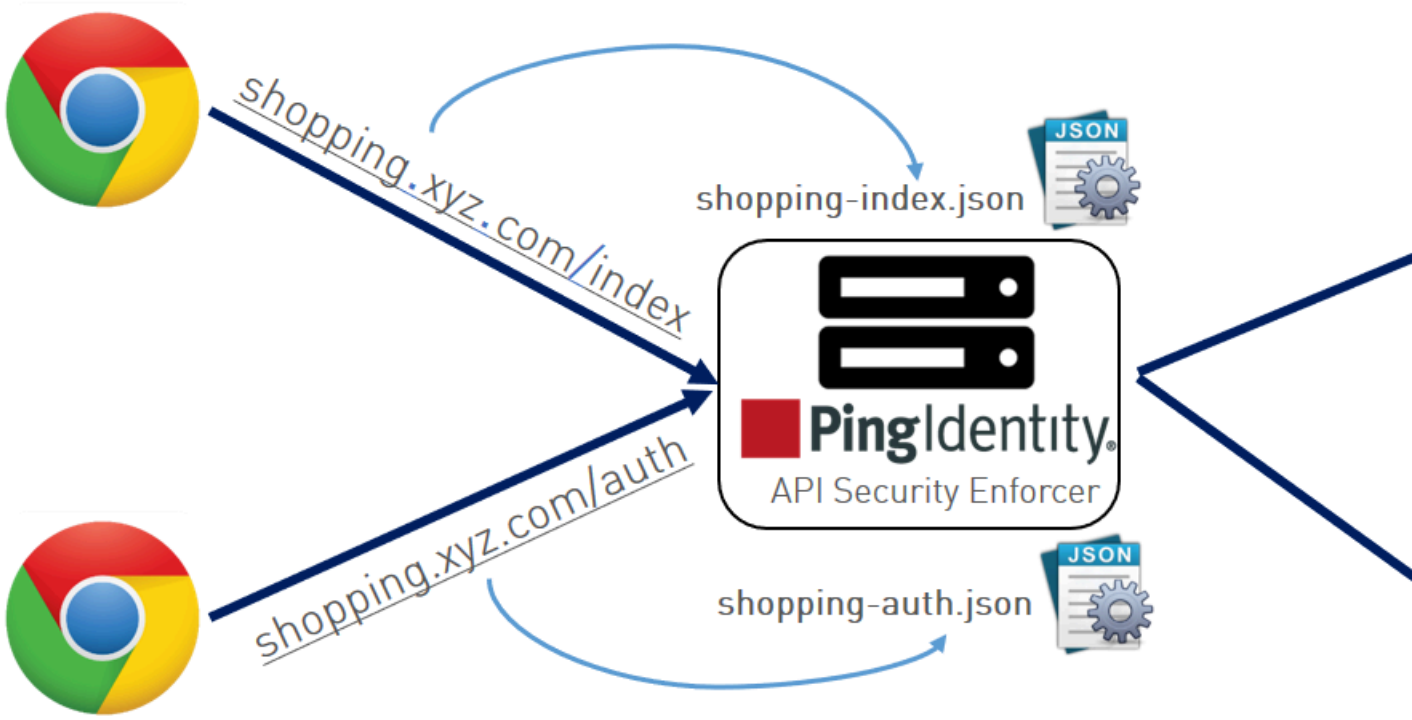
```
"cookie_persistence_enabled": false,
```

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`.



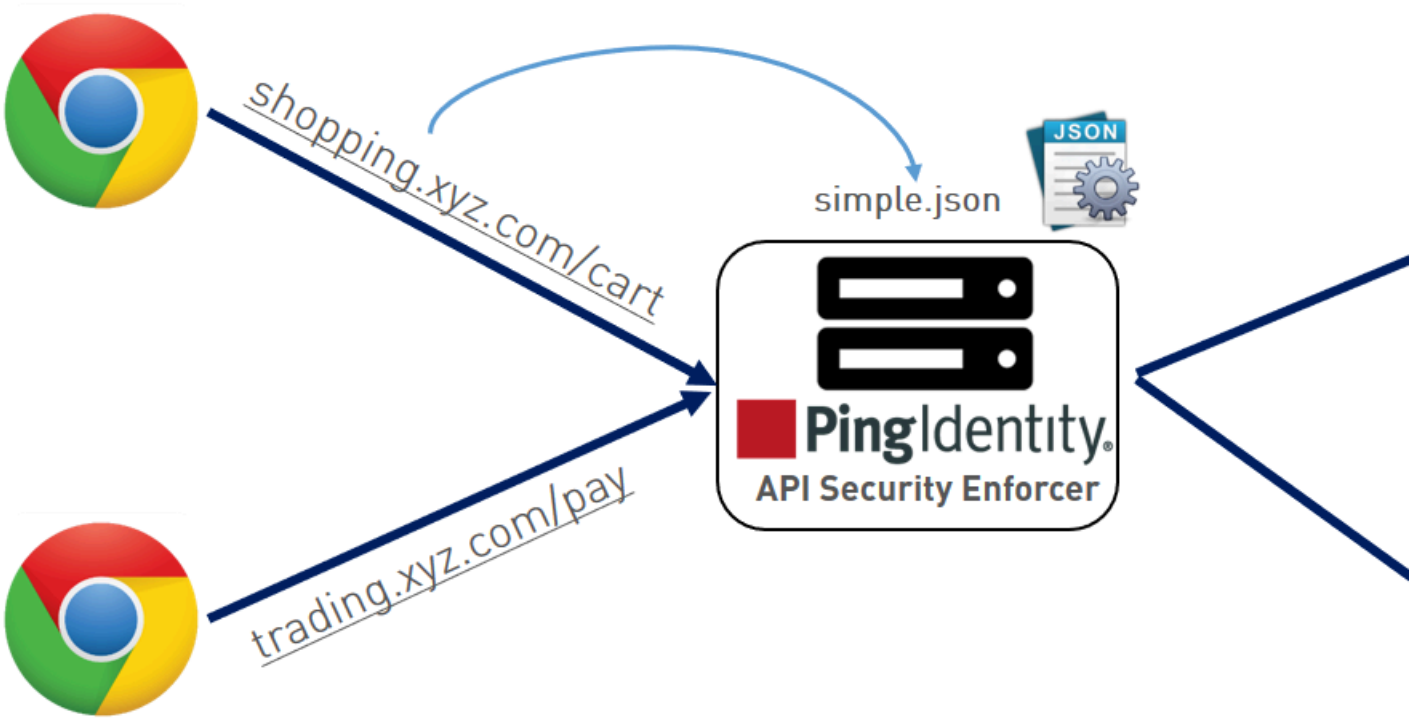
Wildcard hostname and API name

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.



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.

ⓘ **Note:** 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.

Real-time API cybersecurity

API Security Enforcer provides real-time API cybersecurity to stop hackers. Violations are immediately blocked, and attack information is sent to the ABS engine. Real time API Cyber Security is activated only when ASE firewall is enabled.

Enable API cybersecurity

To enable API security, enter the following command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
Firewall is now enabled
```

After enabling API Security, enter the following CLI command to verify cybersecurity is enabled:

```
/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
```

Disable API cybersecurity

To disable ASE's cybersecurity feature, type the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_firewall
Firewall is now disabled
```

After disabling ASE's cybersecurity feature, enter the following CLI command to verify that cybersecurity is disabled:

```
/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 supports the following real time ASE attack detection and blocking:

- **API pattern enforcement** – validate traffic to ensure it is consistent with the API definition
- **API deception** – blocks hackers probing a decoy API (see *API deception environment* on page 218)

Enable ASE detected attacks

Enable real-time ASE attack detection by running the following ASE command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin
 enable_ase_detected_attack
ASE detected attack is now enabled
```

Disable ASE detected attacks

Disable real-time ASE detected attacks by running 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 blacklist.

Configure pattern enforcement

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 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:

- Error_code – for example, “401”
- 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 protocolparameter 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 engine for further analysis and reporting.

API name mapping – hide internal URLs

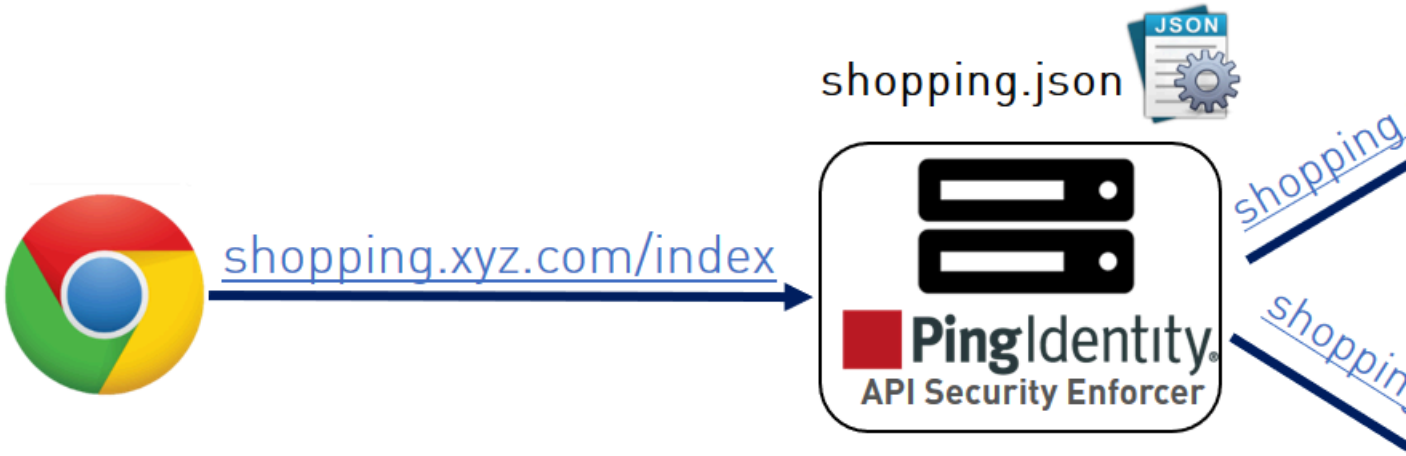
After enabling API cybersecurity, API name mapping can be configured to protect API servers by hiding internal URLs from the outside world. Internal URLs may also be modified without updating entries in the public DNS server.

For example, the following 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 ASE:



Capturing client identifiers

ASE identifies attackers for HTTP(s) and WS(s) protocols using four client identifiers:

- OAuth2 token
- Cookie
- IP address
- API keys

- Username

> ⓘ **Note:** Username is not configured in the api_metadata object of API JSON. However, ASE supports the extraction of usernames coming in a JSON Web Tokens(JWTs), and a JWT object in API JSON is used to configure username information. For more information, see *Extract user information from JWT in inline mode* on page 207 . For usernames that are not part of the JWTs PingIntelligence 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.

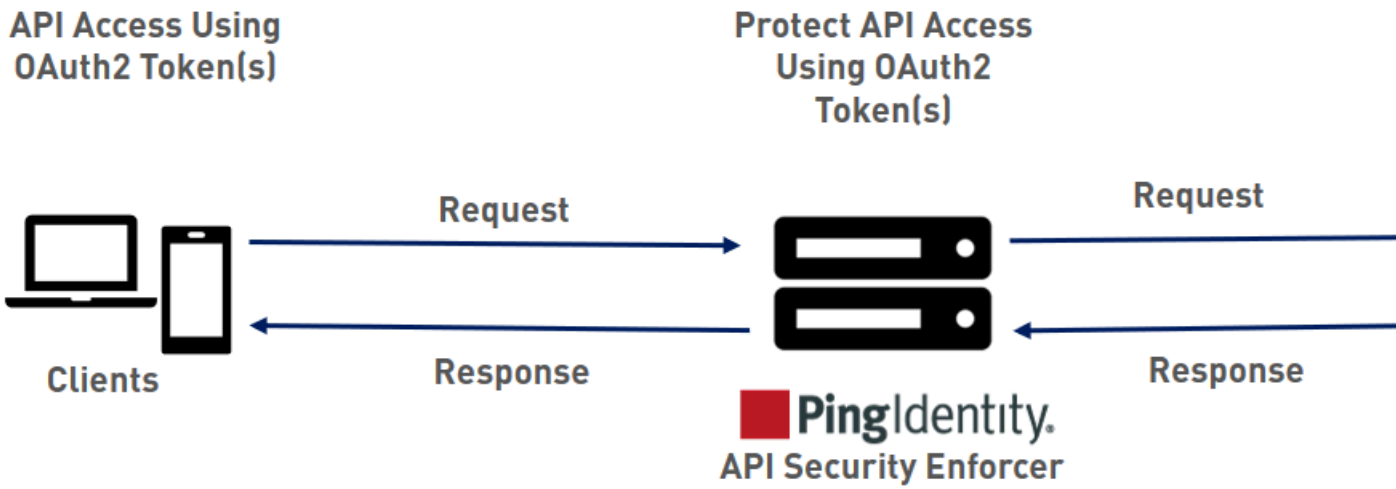
Configure ASE support for OAuth2 tokens

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 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,
 "apikey_qs": "",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": true,
 "api_mapping": {
 "internal_url": ""
 },
```

When blocking is enabled, ASE checks the token against the list of tokens in the whitelist and blacklist. If the token is in the blacklist, the client using the token is immediately blocked.

When pattern enforcement violations are detected on an API configured to support tokens, the attacking client token is added to the blacklist 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:



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).

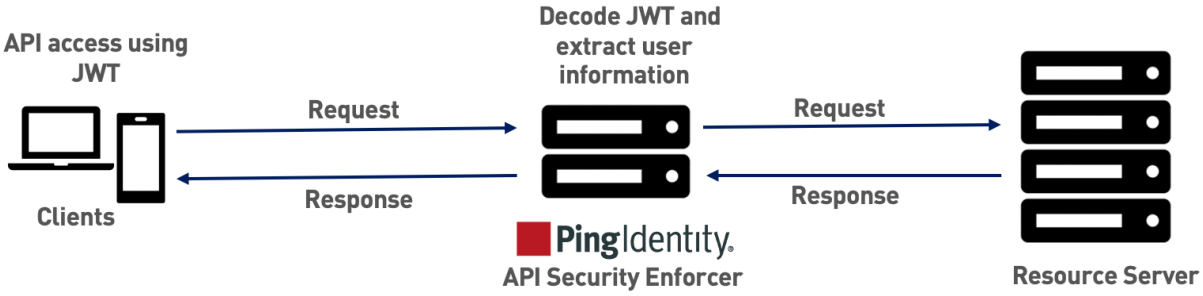
```
"api_metadata": {
 "protocol": "http",
 "url": "/",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "200m",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": true,
 "oauth2_access_token": true,
 "apikey_qs": "key_1.4",
 "apikey_header": "",
 "login_url": "",
 "enable_blocking": 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 Blacklist.

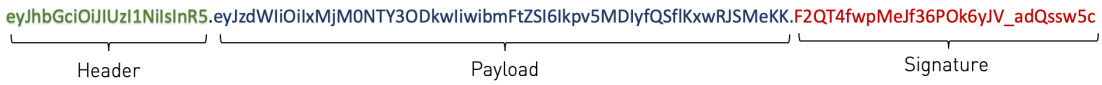
Extract user information from JWT in inline mode

ASE supports the decoding of transparent JSON Web Tokens (JWTs) received as part of API requests. It extracts the user information from the JWT and logs it in ASE access logs. The ABS AI engine analyses these access logs to detect attacks and anomalies.

The following diagram shows the traffic flow when ASE is in inline mode.



A JWT consists of three parts - header, payload, and signature. They are concatenated with periods(**.**). The following is 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 does not validate JWTs. It just decodes the JWTs and extracts the user information.

When ASE is deployed in inline mode, it decodes the JWTs only when 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.

Configure API JSON

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 *Define an Inline API JSON configuration file* on page 194.

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": ""
    },
    "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.

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: eyJhbGciUzI1NiI.eyJzDkwIG4gRG9xpZWQiOjwMjJ9.Dg`
	▪ `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.DWPwNg`
	▪ `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.eyJpc3MiOiJodHRwczot`

username	It is the JWT claim to extract the username.

Parameter	Description
clientid	It is 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 whitelist and blacklist. If the username is in the blacklist, the client using the username is blocked.

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. For more information on API discovery, see *API discovery and configuration* on page 321. 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. It also describes its impact on the APIs discovered by ABS in your environment.

Scenarios	Behavior of ASE	Impact on API discovery
When 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": "" }`
When 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.
When a JWT object is configured with all the keys present but 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.

Manage whitelist and blacklist

ASE maintains the following two types of lists:

- **Whitelist** – 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 CLI commands.
- **Blacklist** – 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:
  - ABS detected attacks (for example data exfiltration). ABS detected attacks have a time-to-live (TTL) in minutes. The TTL is configured in ABS.
  - ASE detected attacks (for example invalid method, decoy API accessed). The ASE detected attacks
  - List of “bad” clients manually generated by CLI

Manage whitelists

Valid operations for OAuth2 Tokens, cookies, IP addresses, API keys, and usernames on a whitelist include:

Add an entry

- Add an IP address to whitelist:

```
/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 whitelist:

```
/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
```

- Add a token to whitelist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist token
 token1.4
token token1.4 added to whitelist
```

- Add an API Key to whitelist:

```
/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
```

- Add a username to whitelist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist username
 abc@example.com
username abc@example.com added to whitelist
```

View whitelist

```
/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
```

Delete an entry

```
/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
```

Clear the whitelist

```
/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
```

Manage blacklists

Valid operations for IP addresses, Cookies, OAuth2 Tokens, and API keys on a blacklist include:

Add an entry

- Add an IP address to blacklist:

```
/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
```

- Add a cookie to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist cookie
 JSESSIONID ad233edqsd1d23redwefew
cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
```

- Add a token to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token
 ad233edqsd1d23redwefew
token ad233edqsd1d23redwefew added to blacklist
```

- Add an API Key to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist api_key
 AccessKey b31dfa4678b24aa5a2daa06aba1857d4
api_key AccessKey b31dfa4678b24aa5a2daa06aba1857d4 added to blacklist
```

- Add an username to blacklist:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist username
 abc@example.com
username abc@example.com added to blacklist
```

> ⓘ **Note:** You can also add username with space to blacklist. For example, "your name".

View blacklist - entire blacklist or based on the type of real time violation.

```
/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
```

Blacklist based on decoy IP addresses

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist decoy
Realtime Decoy Blacklist
1) type : ip, value : 4.4.4.4
```

Blacklist based on protocol violations

```
/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
```

Blacklist based on method violations

```
/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
```

Blacklist based on content-type violation

```
/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
```

ABS detected attacks

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist
 abs_detected
No Blacklist
```

Delete an entry

```
/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
```

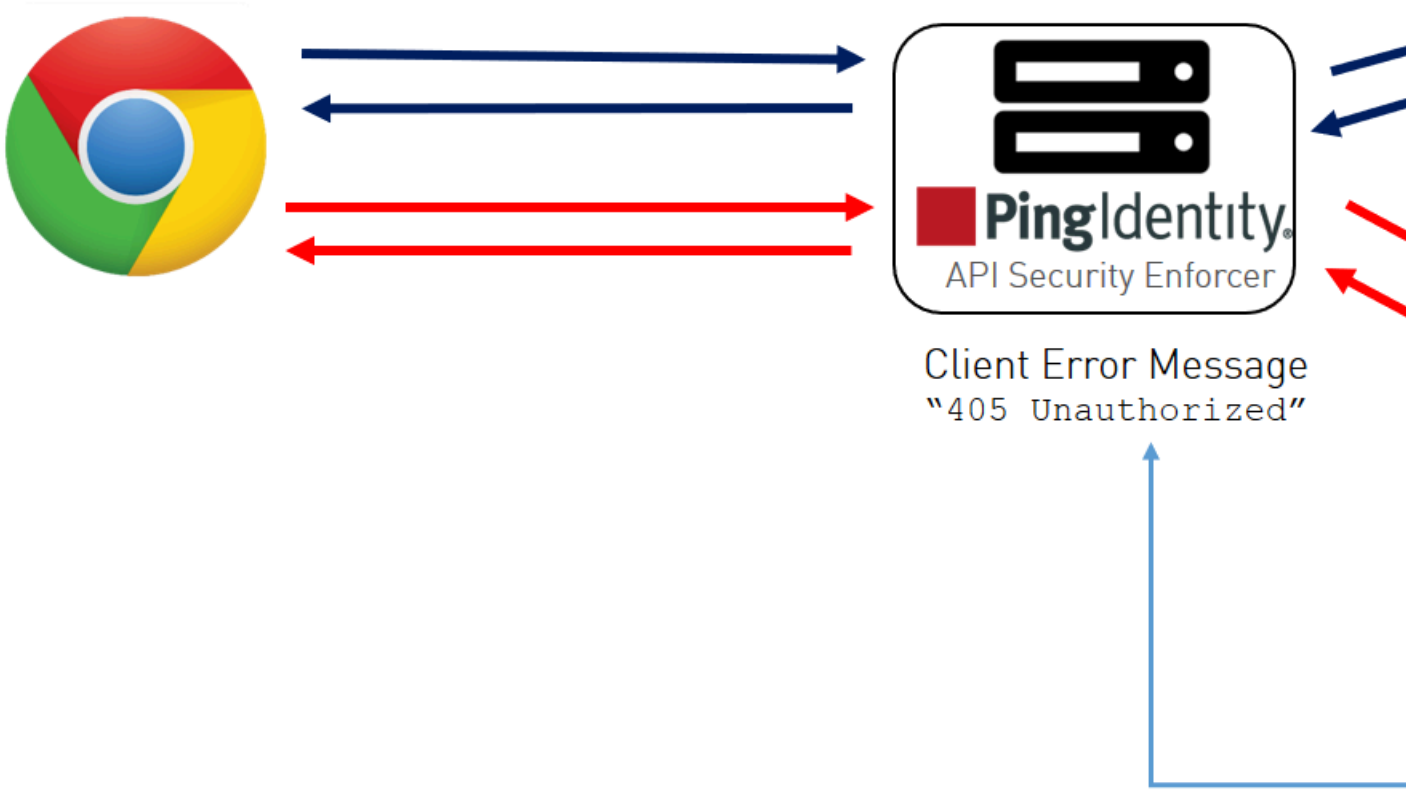
Clear the blacklist

```
./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
```

When clearing the blacklist, make sure that the real-time ASE detected attacks and ABS detected attacks are disabled. If not disabled, the blacklist gets populated again as both ASE and ABS are continuously detecting attacks.

Map server error messages to custom error messages

Backend server error messages (for example, Java stack trace) can reveal internal information to hackers. 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.



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": [
 ]
}
```

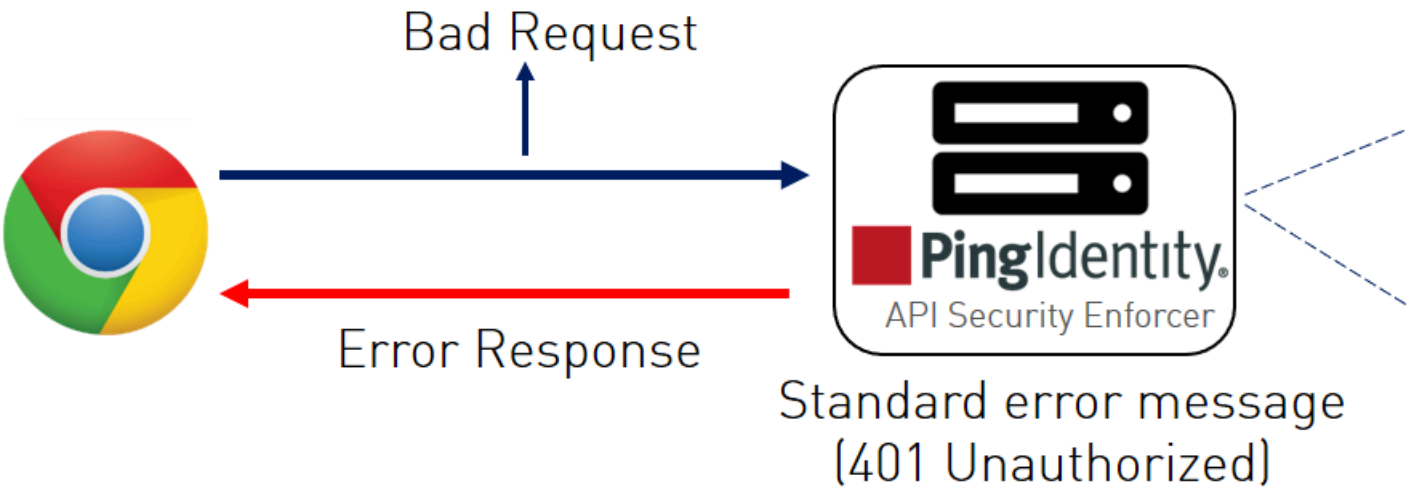
ⓘ **Note:** ASE checks for the presence of the `server_error.json` file. If this file is not available, ASE will not start.

ASE generated error messages for blocked requests

ASE blocks certain requests based on API Mapping or 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 Hostn
Malformed Request	400	Bad Request	Error: Malformed Req
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



Per API blocking

ASE can be configured to selectively block on a per API basis by configuring an API JSON file parameter. To enable per API blocking for each API, set the enable_blocking parameter to `true` in the API JSON. For 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": ""
 },
```

If per API blocking is disabled, ABS still detect attacks for that specific API, however, ASE does not block them. ASE will continue to block attacks on other APIs with the enable_blocking set to `true`.

API deception environment

A decoy API is configured in 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 inresponse_message parameter in API 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.

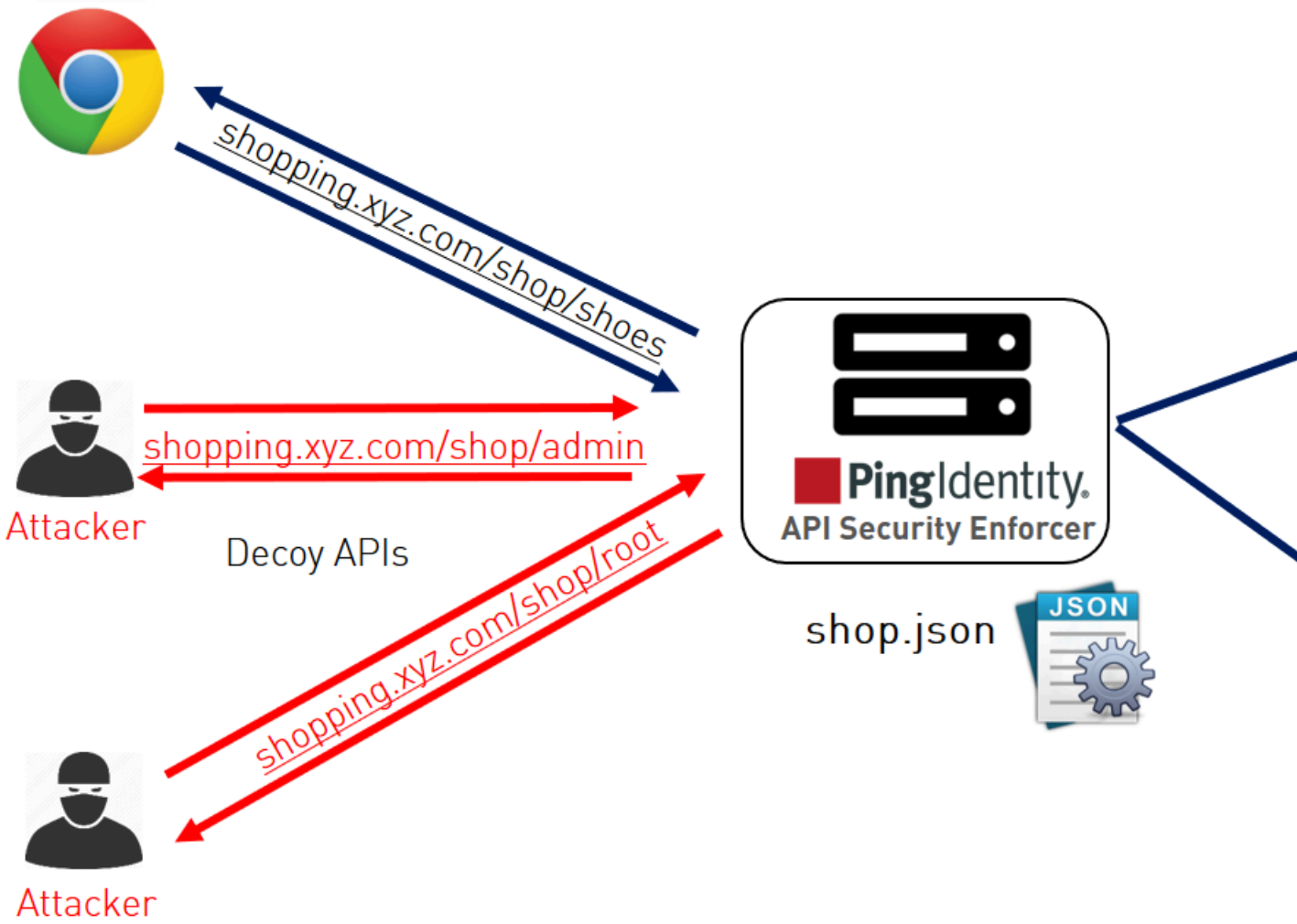
Decoy API traffic is separately logged in files named with the following format: `decoy_pid_<pid_number>__yyyy-dd-mm-<log_file_rotation_time>` (for example, `decoy_pid_8787__2017-04-04_10-57.log)`. decoy log files are rotated every 24-hours and stored in the `opt/pingidentity/ase/logs` directory.

ASE Provides the following decoy API types:

- In-context decoy APIs
- Out-of-context decoy APIs

In-context decoy API

In-context decoy APIs consist of decoy paths within existing APIs supporting legitimate traffic to backend servers. Any traffic accessing a decoy path receives a preconfigured response. For example, in the `shopping` API, `/root` and `/admin` are decoy APIs; `/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 JSON file shows an in-context decoy API:

```
{
 "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:

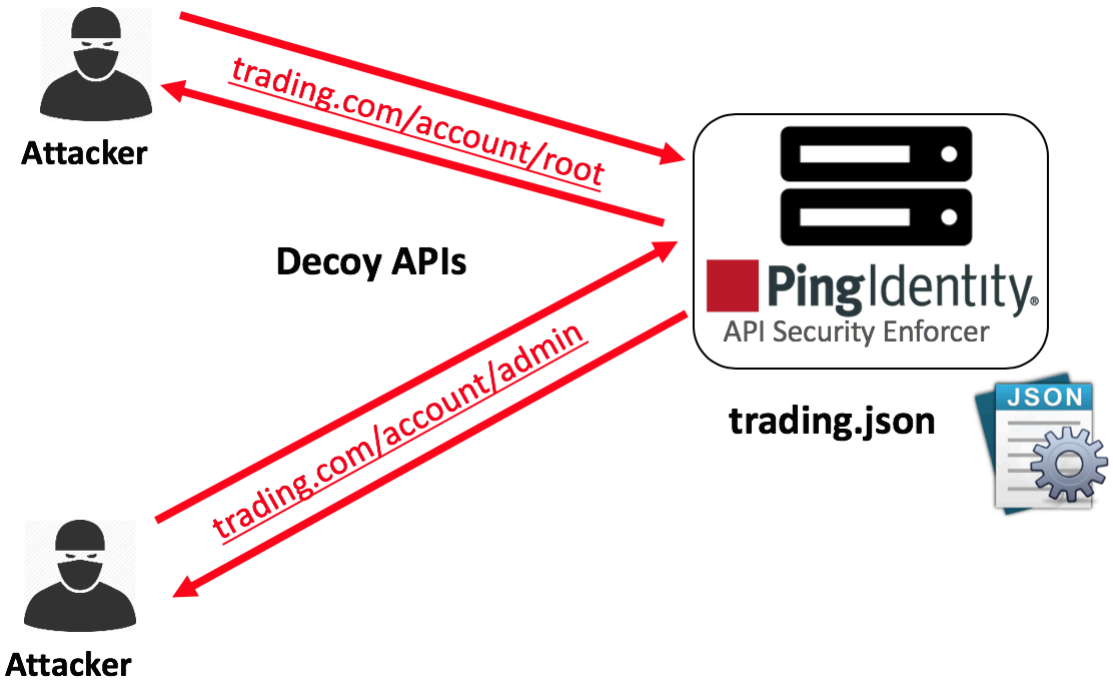
- decoy_enabled parameter is set to true. If set to false, no decoy paths are configured.
- response_code is set to `200`. When a decoy sub-path is accessed, return a `200` response.
- response_def is set to `OK`. When a decoy sub-path is accessed, return `OK` as the response.

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. The figure shows an example.



Following is a snippet of a trading API JSON 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 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
shop ( loaded ), https, decoy: in-context
trading ( loaded ), https, decoy: out-context
```

Real-time API deception attack blocking

ASE detects any client probing a decoy 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 block 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 blacklist 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

ASE flow control ensures that backend 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, Distributed Denial-of-Service (DDoS) attacks).
- **Client Request Queuing** – Queues and retries REST API session requests when servers are busy.

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 WebSock**
Client Spike Threshold	✓	✓
Server Connection Quota	✓	✓
Server Connection Queuing	✓	✓
Server Spike Threshold	✓	-NA-
Bytes-in Threshold	-NA-	✓
Bytes-out Threshold	-NA-	✓

REST API protection from DoS and DDoS

flow control protects REST API servers using four control variables which are independently configured. By default, no flow control is enabled.

Variable	**Description**
	Configured once in every API JSON file
client_spike_threshold	Maximum requests per time-period from a single client IP to a REST API. Time can be in seconds, minutes or hours.
server_connection_queueing	When `true`, queue API connection requests when all backe reach server connection quota. Default value is `false`.
	Configured for each server in every API JSON file

server_connection_quota	Maximum number of concurrent connections to a specific RE a server. Prevents aggregate connections from one or a clus nodes from overloading a REST API running on a specific se
server_spike_threshold	Maximum requests per time-period to the REST API running specified server. Prevents the aggregate request rate from o a cluster of ASE nodes from overloading a REST API runnin specific server. Time can be in seconds, minutes, or hours.

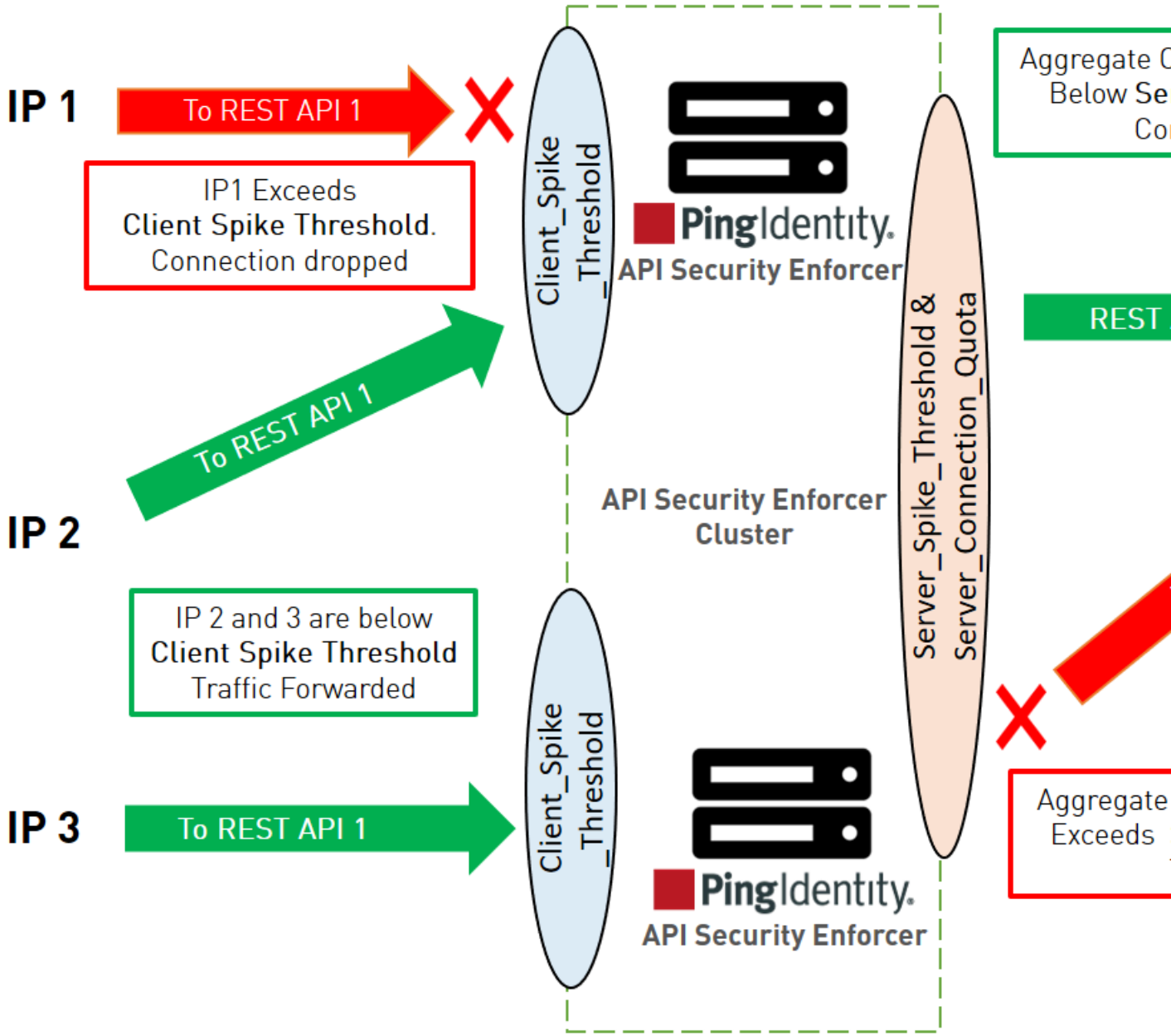
Client flow control monitors incoming traffic from each client connection and drops the session when traffic limits are exceeded. The diagram shows the following client scenarios:

- IP1 sending request volumes which exceed the client_spike_threshold value. ASE 1 sends an error message and terminates the session to stop the attack.
- IP2 and IP3 sending request traffic which 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 including:

- A new connection request from ASE 1 is allowed because it is within the server_connection_quota threshold.
- ASE 2 detects the combined traffic rate from ASE 1 and ASE 2 will exceed the server_spike_threshold for REST API 1. Thus, it drops IP 3 traffic and sends an error message to the client.

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 client_spike_threshold and server-side flow control is regulated by a combination of server_spike_threshold and server_connection_quota.



Example:

Here is an example for an Application Server on the previous diagram.

Variable	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/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.

Server flow control allows 30,000 requests/second to REST API 1 on the application server. If the sum of requests/second from the ASE cluster nodes (i.e. ASE 1 + ASE 2 request rate) to REST API1 exceeds 30,000/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 (i.e. ASE 1 + ASE 2 connection count) to REST API1 exceeds 20,000, then ASE will queue the request for a time since `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 a on a specific server.
server_connection_quota	Maximum number of concurrent sessions from ASE cluster node 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

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": "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 which apply globally across the API definition and include client_spike_threshold and server_connection_queueing. Server specific definitions include server_spike_threshold and server_connection_quota which are 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 and server_spike_threshold.

ⓘ **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.

Flow control CLI for REST 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 server spike threshold

Enter the following command to update the server spike threshold:

```
update_server_spike_threshold {api_id} {host:port} {+ve digit/(second|
minute|hour)}
```

For example: `update_server_spike_threshold shop_api 5000/second`

Update server connection 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 *Enable API security*

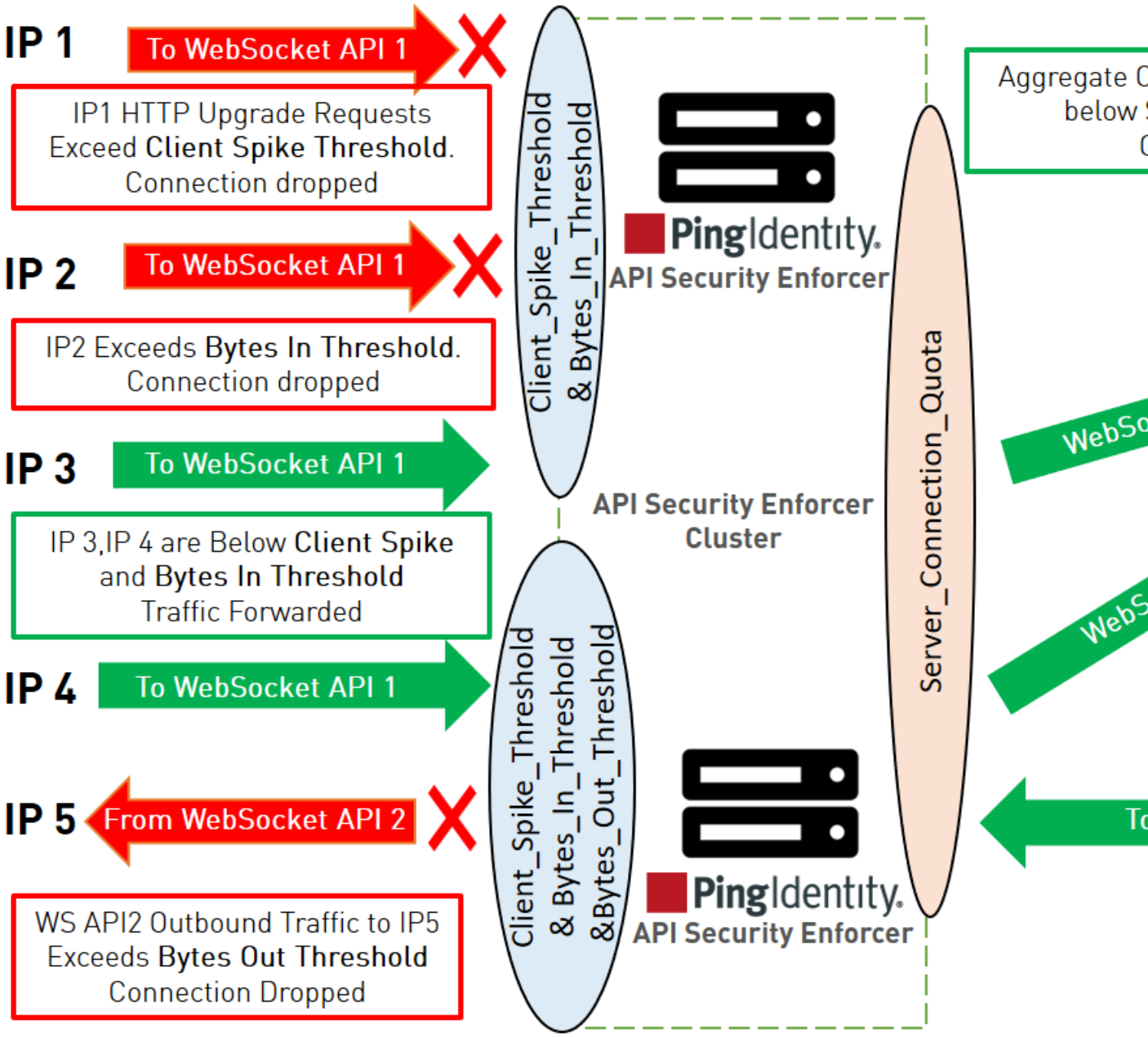
WebSocket API protection from DoS and DDoS

Flow control protects WebSocket servers using five control variables which are independently configured. By default, no flow control is enabled.

Variable	Description
Configured once in every API JSON file	
client_spike_threshold	Maximum number of HTTP requests per time-period from a single 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 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`.
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 client_spike_threshold, bytes_in_threshold, and bytes_out_threshold. The bytes_out threshold protects against data exfiltration. Server flow control is regulated by server_connection_quota.



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.
- 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.

Example:

Here 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
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.

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
server_connection_quota	Maximum number of concurrent sessions from ASE cluster nodes to an API on a specific server.

Configuring flow control for WebSocket API

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..

ⓘ **Note:** API security must be enabled for ASE flow control to work. For more information on enabling API security using the configuration file, see *Define an Inline API JSON configuration file* on page 194 or using the CLI, see *Enable 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 *Enable API Cybersecurity* .

Server connection queuing for REST and WebSocket APIs

ASE can queue server connection requests when the backend API servers are busy. When enabled, server connection queuing applies to both REST and WebSocket APIs and is configured in the API JSON file.

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:

$$\frac{Q_i}{\sum_{i=1}^{n} Q_i}$$

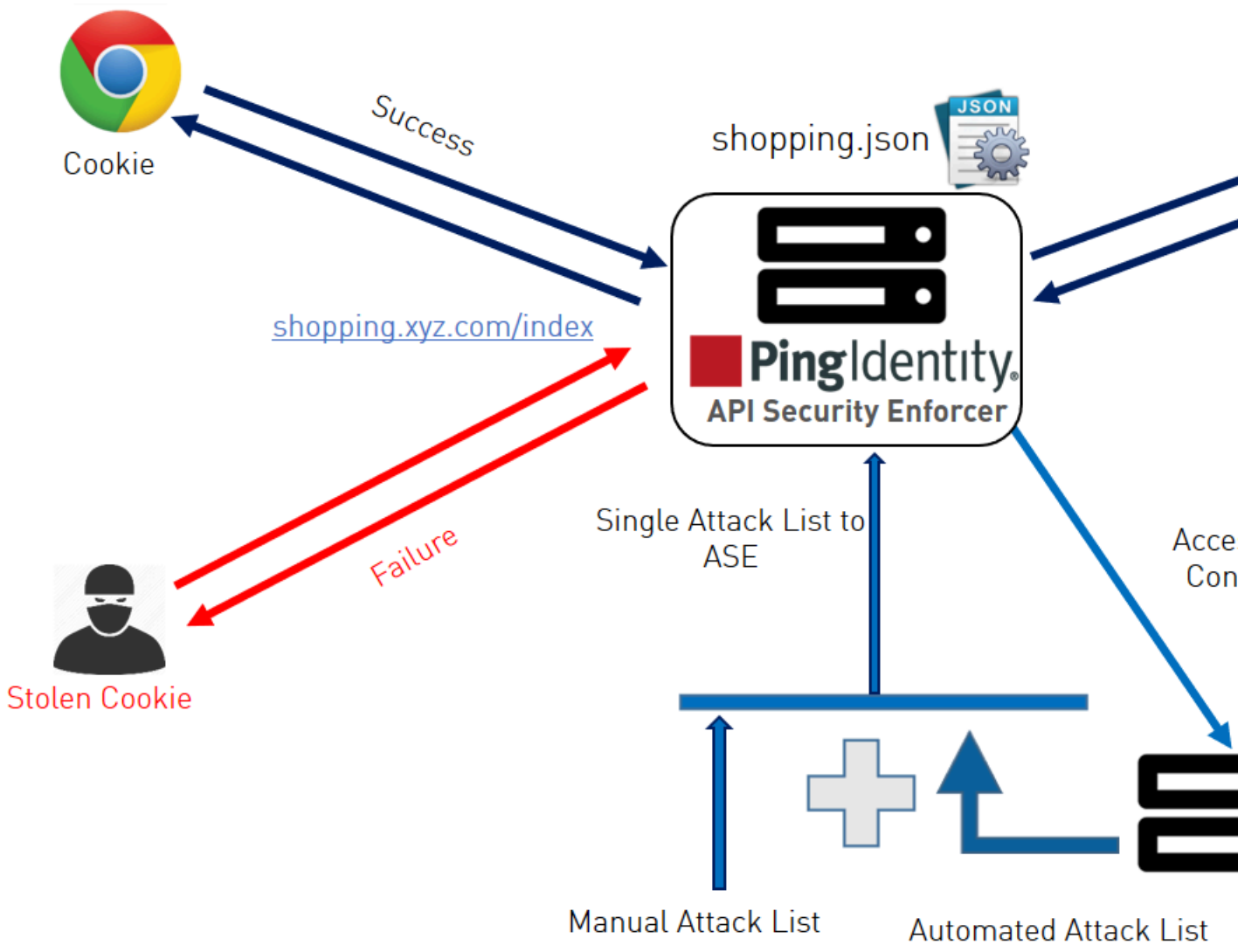
Where *Q i* is the server connection quota for servers from *i=1 to i=n*

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.

ABS AI-based security

ABS AI engine detects attacks using artificial intelligence (AI) algorithms. After receiving ASE access logs and API JSON configuration files, ABS applies AI algorithms to track API connections and detect attacks. If enable_abs_attack is `true`, ABS sends blacklist to ASE which blocks client identifiers, like, API keys, usernames, cookie, IP address, and OAuth token on the list.



Configure ASE to ABS connectivity

To connect ASE to ABS, configure the ABS address (IPv4:Port or Hostname:Port), access key, and secret key in the `abs.conf` file located in the `/opt/pingidentity/ase/config` directory.

ⓘ **Note:** enable_absmust be set to `true` in the `ase.conf` file. when ABS is in a different AWS security group, use a private IP address

The parameter values and descriptions are included in the following table:

Parameter	Description
abs_endpoint	Hostname and port or the IPv4 and port of all the ABS nodes

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 ⓘ **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 the ABS nodes. The same value has to be configured in ABS Mongo database. This value is obfuscated during the start of ASE. ⓘ **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 the value is set to false.
abs_ca_cert_path	Location of the trusted 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.

ⓘ **Note:** The access_key and secret_key are configured in ABS. For more information, see *ABS Admin Guide*.

Here 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.
; 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-ABS encrypted communication

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`. The abs_ca_cert_path is the location of ABS's trusted CA certificate. If the field is left empty, ASE does not verify ABS's certificate, however, the communication is till encrypted.

Check and open ABS ports

The default ports for connection with ABS are 8080 and 9090. Run the **`check_ports_ase.sh`** script on the ASE machine to determine ABS accessibility. Input ABS host IP address and ports as arguments.

```
/opt/pingidentity/ase/util ./check_ports_ase.sh {ABS IPv4:[port]}
```

Manage ASE blocking of ABS detected attacks

To configure ASE to automatically fetch and block ABS detected attacks, complete the following steps:

1. Enable ASE Security. Enter the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_firewall
   ```

2. Enable ASE to send API traffic information to ABS. Enter the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs
   ```

3. Enable ASE to fetch and block ABS detected attacks. Enter the following command:

   ```
   /opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs_attack
   ```

After enabling automated attack blocking, ASE periodically fetches the attack list from ABS and blocks the identified connections. 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.

```
; This value determines how often ASE will query ABS.
abs_attack_request_minutes=10
```

Disable attack list fetching from ABS

To disable ASE from fetching the ABS attack list, entering the following CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs_attack
```

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. 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 Admin Guide.*

To stop an ASE cluster from sending log files to ABS, enter the following ASE CLI command.

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs
```

After entering this command, ABS will not receive any logs from ASE. Refer to the ABS documentation for information on types of attacks.

CLI for inline ASE

Start ASE

Description

Starts ASE

Syntax

`./start.sh`

Stop ASE

Description

Stops ASE

Syntax

```
./stop.sh
```

Help

Description

Displays cli.sh help

Syntax

```
./cli.sh help
```

Version

Description

Displays the version number of ASE

Syntax

```
./cli.sh version
```

Status

Description

Displays the running status of ASE

Syntax

```
./cli.sh status
```

Update Password

Description

Change ASE admin password

Syntax

```
./cli.sh update_password {-u admin}
```

Change log level

Description

Change balancer.log and controller.log log level

Syntax

```
./cli.sh log_level -u admin -p
```

options - `warn`, `info`, `error`, `fatal`, `debug`

Get Authentication Method

Description

Display the current authentication method

Syntax

```
./cli.sh get_auth_method {method} {-u admin}
```

Update Authentication Method

Description

Update ASE authentication method

Syntax

```
./cli.sh update_auth_method {method} {-u admin}
```

Enable Audit Logging

Description

Enable audit logging

Syntax

```
./cli.sh enable_audit -u admin -p admin
```

Disable Audit Logging

Description

Disable audit logging

Syntax

```
./cli.sh disable_audit -u admin -p admin
```

Add Syslog Server

Description

Add a new syslog server

Syntax

```
./cli.sh –u admin -p admin add_syslog_server host:port
```

Delete Syslog Server

Description

Delete the syslog server

Syntax

```
./cli.sh –u admin -p admin delete_syslog_server host:port
```

List Syslog Server

Description

List the current syslog server

Syntax

```
./cli.sh –u admin -p admin list_syslog_server
```

Add API

Description

Add a new API from config file in JSON format. File should have `.json` extension

Syntax

```
./cli.sh –u admin -p admin add_api {config_file_path}
```

Update API

Description

Update an API after the API JSON file has been edited and saved.

Syntax

```
./cli.sh –u admin -p admin update_api {api_name}
```

List APIs

Description

Lists all APIs configured in ASE

Syntax

```
./cli.sh –u admin -p admin list_api
```

API Info

Description

Displays the API JSON file

Syntax

```
./cli.sh –u admin –p admin api_info {api_id}
```

API Count

Description

Displays the total number of APIs configured

Syntax

```
./cli.sh –u admin –p admin api_count
```

List API Mappings

Description

Lists all the external and internal URL mappings.

Syntax

```
./cli.sh –u admin –p admin list_api_mappings
```

Delete API

Description

Delete an API from ASE. Deleting an API removes the corresponding JSON file and deletes all the cookies associated with that API

Syntax

```
./cli.sh –u admin –p admin delete_api {api_id}
```

Add a Server

Description

Add a backend server to an API. Provide the IP address and port number of the server

Syntax

```
./cli.sh –u admin –p admin add_server {api_id}{host:port}[quota]
[spike_threshold]
```

List Server

Description

List all servers for an API

Syntax

```
./cli.sh –u admin –p admin list_server {api_id}
```

Delete a Server

Description

Delete a backend server from an API. Provide the IP address and port number of the server

Syntax

```
./cli.sh –u admin –p admin delete_server {api_id}{host:port}
```

Enable Per API Blocking

Description

Enables attack blocking for the API

Syntax

```
./cli.sh –u admin –p admin enable_blocking {api_id}
```

Disable Per API Blocking

Description

Disable attack blocking for the API

Syntax

```
./cli.sh –u admin –p admin disable_blocking {api_id}
```

Enable Health Check

Description

Enable health check for a specific API

Syntax

```
./cli.sh -u admin -p admin enable_health_check shop_api
```

Disable Health Check

Description

Disable health check for a specific API

Syntax

```
./cli.sh –u admin –p admin disable_health_check {api_id}
```

Generate Master Key

Description

Generate the master obfuscation key ase_master.key

Syntax

```
./cli.sh –u admin –p admin generate_obfkey
```

Obfuscate Keys and Password

Description

Obfuscate the keys and passwords configured in various configuration files

Syntax

```
./cli.sh –u admin –p admin obfuscate_keys
```

Create a Key Pair

Description

Creates private key and public key pair in keystore

Syntax

```
./cli.sh –u admin –p admin create_key_pair
```

Create a CSR

Description

Creates a certificate signing request

Syntax

```
./cli.sh –u admin –p admin create_csr
```

Create a Self-Signed Certificate

Description

Creates a self-signed certificate

Syntax

```
./cli.sh –u admin –p admin create_self_sign_cert
```

Import Certificate

Description

Import CA signed certificate into keystore

Syntax

```
./cli.sh –u admin -p admin import_cert {cert_path}
```

Create Management Key Pair

Description

Create a private key for management server

Syntax

```
/cli.sh –u admin -p admin create_management_key_pair
```

Create Management CSR

Description

Create a certificate signing request for management server

Syntax

```
/cli.sh –u admin -p admin create_management_csr
```

Create Management Self-signed Certificate

Description

Create a self-signed certificate for management server

Syntax

```
/cli.sh –u admin -p admin create_management_self_sign_cert
```

Import Management Key Pair

Description

Import a key-pair for management server

Syntax

```
/cli.sh –u admin -p admin import_management_key_pair {key_path}
```

Import Management Certificate

Description

Import CA signed certificate for management server

Syntax

```
/cli.sh –u admin -p admin import_management_cert {cert_path}
```

Health Status

Description

Displays health status of all backend servers for the specified API

Syntax

```
./cli.sh –u admin -p admin health_status {api_id}
```

Cluster Info

Description

Displays information about an ASE cluster

Syntax

```
./cli.sh –u admin -p admin cluster_info
```

Server Count

Description

Lists the total number of APIs associated with an API

Syntax

```
./cli.sh –u admin -p admin server_count {api_id}
```

Cookie Count

Description

Lists the live cookie count associated with an API

Syntax

```
./cli.sh –u admin -p admin cookie_count {api_id}
```

Persistent Connection Count

Description

Lists the WebSocket or http-keep alive connection count for an API

Syntax

```
./cli.sh –u admin -p admin persistent_connection_count {api_id}
```

Clear cookies

Description

Clear all cookies for an API

Syntax

```
./cli.sh –u admin -p admin clear_cookies{api_id}
```

Enable Firewall

Description

Enable API firewall. Activates pattern enforcement, API name mapping, manual attack type

Syntax

```
./cli.sh –u admin -p admin enable_firewall
```

Disable Firewall

Description

Disable API firewall

Syntax

```
./cli.sh –u admin -p admin disable_firewall
```

Enable ASE detected attacks

Description

Enable ASE detected attacks

Syntax

```
./cli.sh –u admin -p admin enable_ase_detected_attack
```

Disable ASE Detected Attacks

Description

Disable API firewall

Syntax

```
./cli.sh –u admin -p admin disable_ase_detected_attack
```

Enable ABS

Description

Enable ABS to send access logs to ABS

Syntax

```
./cli.sh –u admin -p admin enable_abs
```

Disable ABS

Description

Disable ABS to stop sending access logs to ABS

Syntax

```
./cli.sh –u admin –p admin disable_abs
```

Enable ABS Detected Attack Blocking

Description

Enable ASE to fetch ABS detected attack lists and block access of list entries.

Syntax

```
./cli.sh –u admin –p admin enable_abs_attack
```

Disable ABS Detected Attack Blocking

Description

Stop ASE from blocking and fetching ABS detected attack list. This command does not stop ABS from detecting attacks.

Syntax

```
./cli.sh –u admin –p admin disable_abs_attack
```

Adding Blacklist

Description

Add an entry to ASE blacklist using CLI. Valid type values are: IP, Cookie, OAuth2 token, API Key, and username

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

Syntax

```
./cli.sh –u admin –p admin add_blacklist {type}{name}{value}
```

Example

/cli.sh -u admin -p admin add_blacklist ip 1.1.1.1

Delete Blacklist Entry

Description

Delete entry from the blacklist.

Syntax

```
./cli.sh –u admin –p admin delete_blacklist {type}{name}{value}
```

Example

cli.sh -u admin -p delete_blacklist token 58fcb0cb97c54afbb88c07a4f2d73c35

Clear Blacklist

Description

Clear all the entries from the blacklist

Syntax

```
./cli.sh –u admin –p admin clear_blacklist
```

View Blacklist

Description

View the entire blacklist or view a blacklist for the specified attack type (for example, invalid_method)

Syntax

```
./cli.sh –u admin –p admin view_blacklist {all|manual|abs_generated|
invalid_content_type|invalid_method|invalid_protocol|decoy}
```

Adding Whitelist

Description

Add an entry to ASE whitelist using CLI. Valid type values are: IP, cookie, OAuth2 token, API key, and username

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

Syntax

```
./cli.sh –u admin –p admin add_whitelist {type}{name}{value}
```

Example

/cli.sh -u admin -p admin add_whitelist api_key AccessKey 065f73cdf39e486f9d7cda97d2dd1597

Delete Whitelist Entry

Description

Delete entry from the whitelist

Syntax

```
./cli.sh –u admin –p admin delete_whitelist {type}{name}{value}
```

Example

/cli.sh -u admin -p delete_whitelist token 58fcb0cb97c54afbb88c07a4f2d73c35

Clear Whitelist

Description

Clear all the entries from the whitelist

Syntax

```
./cli.sh –u admin –p admin clear_whitelist
```

View Whitelist

Description

View the entire whitelist

Syntax

```
./cli.sh –u admin –p admin view_whitelist
```

ABS Info

Description

Displays ABS status information.

ABS enabled or disabled, ASE fetching ABS attack types, and ABS cluster information

Syntax

```
./cli.sh –u admin –p admin abs_info
```

Enable XFF

Description

Enable X-Forwarded For

Syntax

```
./cli.sh –u admin –p admin enable_xff
```

Disable XFF

Description

Disable X-Forwarded For

Syntax

```
./cli.sh –u admin –p admin disable_xff
```

Update Client Spike

Description

Update Client Spike Threshold

Syntax

```
update_client_spike_threshold {api_id} {+ve digit/(second|minute|hour)}
```

Example

update_client_spike_threshold shop_api 5000/second

Update Server Spike

Description

Update Server Spike Threshold

"*" - use the same value for all servers

Syntax

```
update_server_spike_threshold {api_id} {host:port} {+ve digit/(second|
minute|hour)}
```

Example

update_server_spike_threshold shop_api 127.0.0.1:9090 5000/second

update_server_spike_threshold shop_api "*" 5000/second

Update Bytes-in

Description

Update bytes in value for a WebSocket API

Syntax

```
update_bytes_in_threshold {api_id} {+ve digit/(second|minute|hour)}
```

Example

update_bytes_in_threshold shop_api 8096/second

Update Bytes-out

Description

Update bytes out value for a WebSocket API

Syntax

```
update_bytes_out_threshold {api_id} {+ve digit/(second|minute|hour)}
```

Example

update_bytes_out_threshold shop_api 8096/second

Update Server Quota

Description

Update the number of API connections allowed on a backend server

"*" - use the same value for all backend servers

Syntax

```
update_server_connection_quota {api_id} {host:port} {+ve digit}
```

Example

update_server_connection_quota shop_api 127.0.0.1:9090 5000

update_server_connection_quota shop_api "*" 5000

ASE REST APIs using Postman

Multiple options are available for accessing the ASE REST API reporting including:

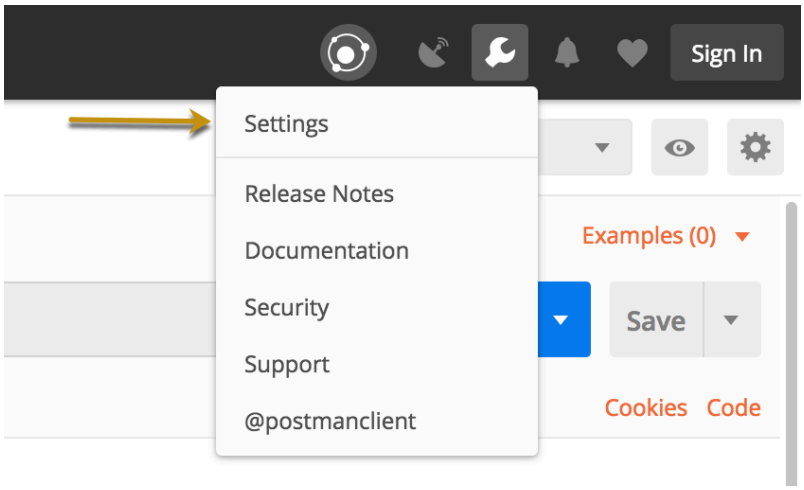
- Postman App
- Java, Python, C Sharp, or similar languages.
- Java client program (such as Jersey)
- C sharp client program (such as RestSharp)

For the Postman application, Ping Identity provides two set of Postman collections which are used by Postman to access the ASE REST API JSON information. The collections for Inline and Sideband ASE. Make sure to install Postman 6.2.5 or higher.

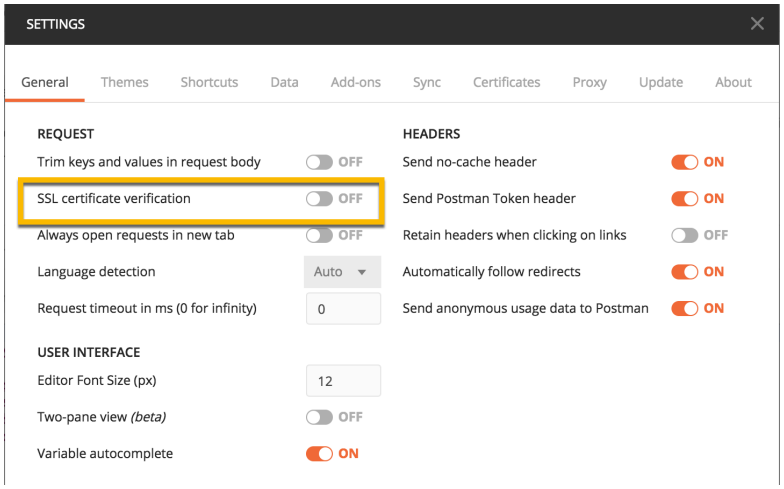
ASE self-signed certificate with Postman

ASE ships with a self-signed certificate. If you want to use Postman with the self-signed certificate of ASE, then from Postman's settings, disable the certificate verification option. Complete the following steps to disable Postman from certificate verification:

1. Click on the **spanner**  on the top-right corner of Postman client. A drop-down window is displayed.
2. Select **Settings** from the drop-down window:



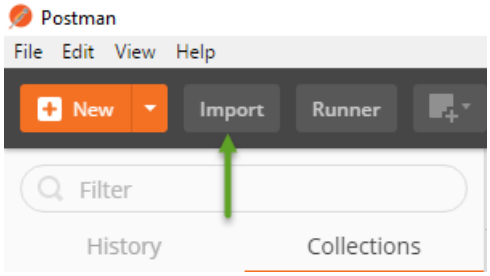
3. In the Settings window, switch-off certificate verification by clicking on the SSL certificate verification button:



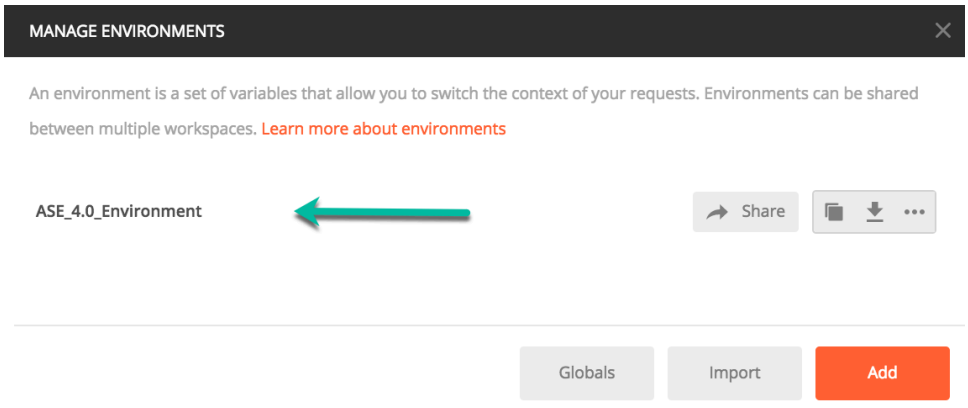
View ASE REST APIs in Postman

To view the reports, complete the following steps:

1. Download `ASE_4.0_Inline` or `ASE_4.0_Sideband` and `ASE_4.0_Environment` JSON files from 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.

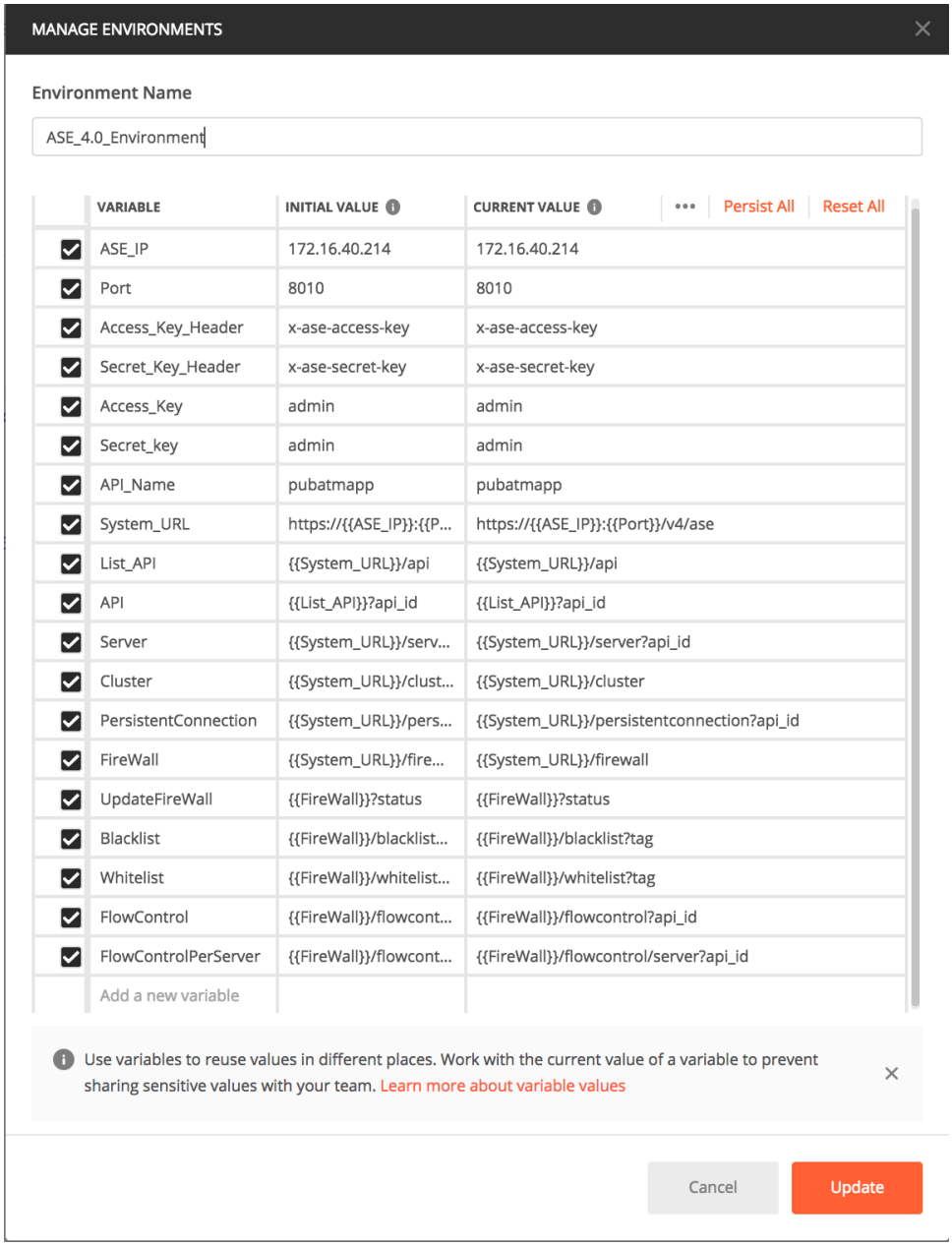


4. After importing the files, click the gear  button in the upper right corner.
5. In the **MANAGE ENVIRONMENTS** pop-up window, click **ASE_4.0_Environment**



6. In the pop-up window, configure the following values and then click **Update**
   - **ASE_IP**: IP address of the ASE node.
   - **Port**: Port number of the ASE node.
   - **Access_Key_Header** and **Secret_Key_Header**: Use the default values.
   - **Access_Key** and **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`.



7. In the main Postman window, select the report to display on the left column and then click **Send**.

REST API for inline and sideband ASE

ASE REST API allows you to manage adding, removing, and modifying your backend servers. The REST API payload uses a JSON format. REST API also helps in integrating 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 Blacklist (POST)* – Inline and sideband ASE
- *Delete Attack Type from the Whitelist (DELETE)* – Inline and sideband ASE
- *Clear the Blacklist (DELETE)* – Inline and sideband ASE
- *View Blacklist (GET)* – Inline and sideband ASE
- *Add Attack Type to Whitelist (POST)* – Inline and sideband ASE
- *Delete Attack Type from the Whitelist (DELETE)* – Inline and sideband ASE
- *Clear Whitelist (DELETE)* – Inline and sideband ASE
- *View Whitelist (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

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

```
{
 "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": "",
 "response_message": "",
 "decoy_subpaths": []
 }
 }
}
```

WebSocket API request

```
{
 "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": "",
 "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” :”ap 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

Response

HTTP Code	Status	Content body (application/json)

200	success	**REST API**
		{
		"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": "",
		"response_message": "",
		"decoy_subpaths": []
		}
		}
		}
		WebSocket API

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>

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

<pre>{
"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": ""</pre>

```
 },
 "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": "",
 "response_message": "",
 "decoy_subpaths": []
 }
 }
}
```

WebSocket API request

```
{
 "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": "",
 "response_message": "",
 "decoy_subpaths": []
 }
 }
}
```

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”}

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”}

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”}
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

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>
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
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

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>
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 blacklist (POST)

Request

POST	/v4/ase/firewall/blacklist
x-ase-access-key	<Access Key>
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 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 blacklist (DELETE)

Request

DELETE	/v4/ase/firewall/blacklist
x-ase-access-key	<Access Key>
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 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 blacklist (DELETE)

Request

DELETE	/v4/ase/firewall/blacklist?tag=all
x-ase-access-key	<Access Key>
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 blacklist (GET)

Request

GET	/v4/ase/firewall/blacklist?tag=
Tags	tag=all (default is 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

Response

Status code	Response body

200 OK	<pre>{
	"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",
	}
	]
	}</pre>

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 whitelist (POST)

Request

POST	/v4/ase/firewall/whitelist
x-ase-access-key	<Access Key>
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

Delete attack type from the whitelist (DELETE)

Request

DELETE	/v4/ase/firewall/whitelist
x-ase-access-key	<Access Key>
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 whitelist (DELETE)

Request

DELETE	/v4/ase/firewall/whitelist?tag=all
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 whitelist (POST)

Request

GET	/v4/ase/firewall/whitelist
x-ase-access-key	<Access Key>
x-ase-secret-key	<Secret Key>

Accept	application/json

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

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

```
{ "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"}`

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" }`
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 entries in the audit log files have four components as shown in the following table:

Date	Subject	Action	Resources
`YYYY-MM-DD hh:mm:ss`	Subject is the module through which actions are performed: CLI, REST API or cluster	Actions are the executed commands.	Resources parameters the actions.

Following are the subjects and their description:

Subject	Description
`cli`	CLI commands executed
`rest_api`	REST API requests received by 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

Action	Resources
status	-NA-
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=
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	-NA-
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	-SAME AS ABOVE-
DELETE /v4/ase/api	-SAME AS ABOVE-
POST /v4/ase/server	-SAME AS ABOVE-
GET /v4/ase/server	-SAME AS ABOVE-
DELETE /v4/ase/server	-SAME AS ABOVE-
GET /v4/ase/cluster	-SAME AS ABOVE-
POST /v4/ase/firewall	-SAME AS ABOVE-
GET /v4/ase/firewall	-SAME AS ABOVE-
POST /v4/ase/firewall/flowcontrol	-SAME AS ABOVE-
GET /v4/ase/firewall/flowcontrol	-SAME AS ABOVE-
POST /v4/ase/firewall/flowcontrol/server	-SAME AS ABOVE-

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=
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

A complete list of supported encryption protocols for TLS1.2 based on the operating system is shown in the boxes below.

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
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
ECDHE-RSA-AES128-SHA256	DH-RSA-AES256-SHA256
ECDHE-ECDSA-AES128-SHA256	DH-DSS-AES256-SHA256

Autoscaling ASE in AWS environment

You can auto-scale ASE setup in AWS environment by completing the following steps:

1. Create and AMI for ASE
2. Create an IAM role in the Security, Identity, and Compliance
3. Create the Security Group
4. Create Launch Configuration
5. Create an Autoscale group

Create an AMI for ASE

Complete the following steps to create an AMI for ASE:

1. Create an RHEL 7.6 or Ubuntu 16.04 LTS EC2 instance

2. Install the AWS CLI by completing the following steps:
   a. Install Python 2.7
   b. Enter the following command:

```
sudo curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o
 "awscli-bundle.zip"
```

   c. Unzip the CLI bundle

```
sudo unzip awscli-bundle.zip
```

   d. 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. Untar the binary in the EC2 instance. At the command prompt, type the following command to untar the ASE file:

```
tar –zxvf <filename>
```

   For example:

   `tar –zxvf ase-rhel-4.0.tar.gz`
5. To verify that ASE successfully installed, enter the ls command at the command prompt. This should list the pingidentity directory and the build's tar file.**For example:**

```
/opt/$ ls
pingidentity ase-rhel-4.0.tar.gz
```

6. Change 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.

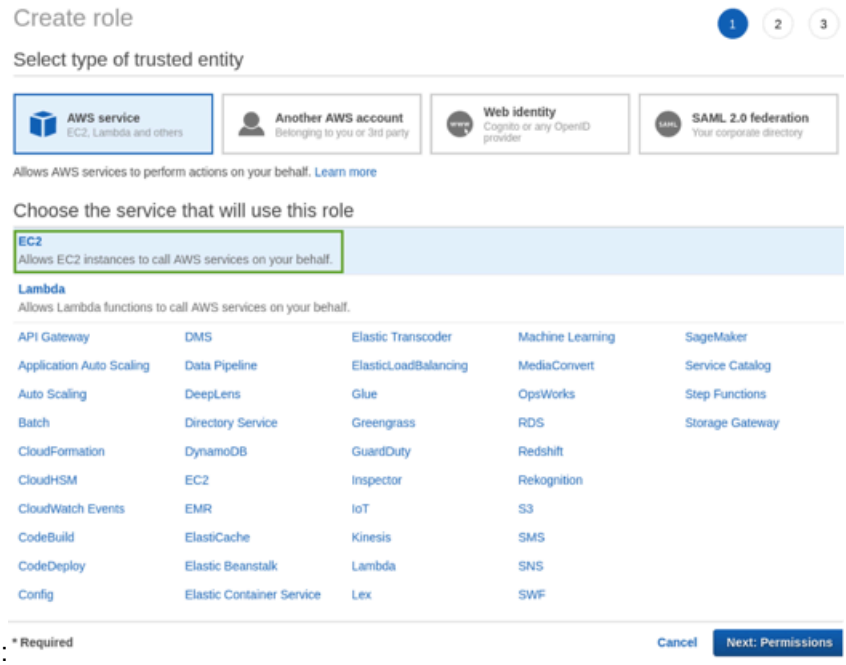
> ⓘ **Note:** When you are creating the AMI, do not select the “No Reboot” option

Create an IAM role in the security, identity, and compliance

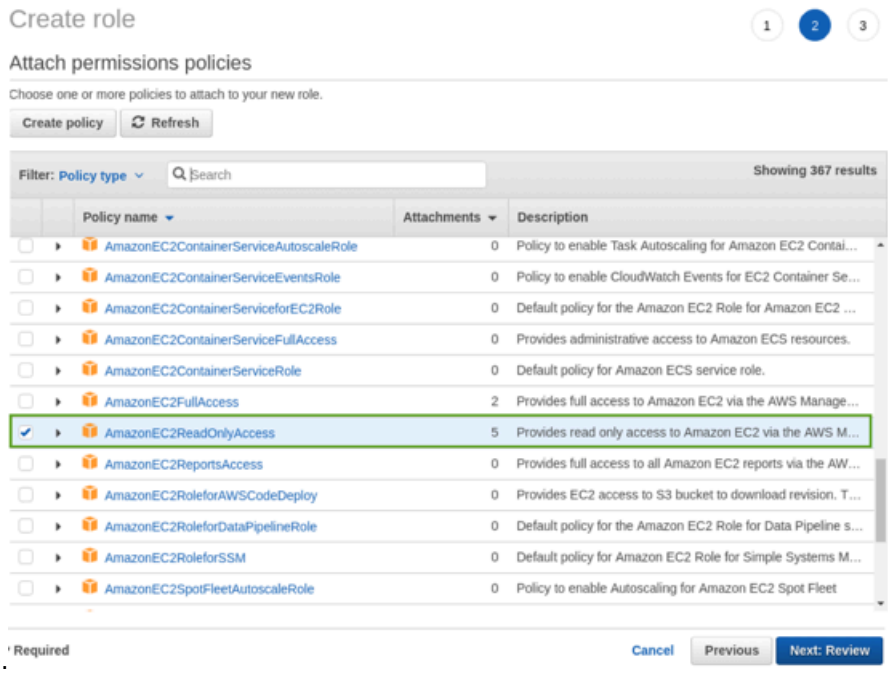
About this task
Complete the following steps to create an IAM role in the security, identity, and compliance:

Steps

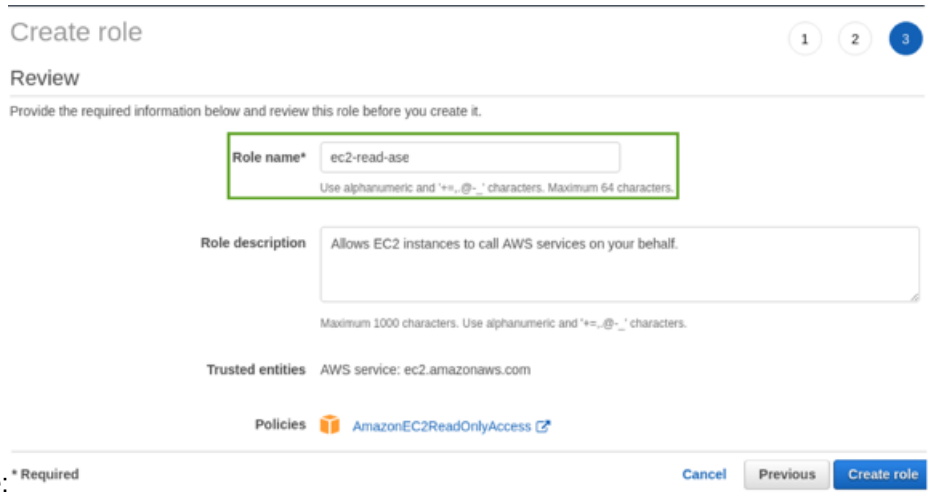
1. Create an IAM role by selecting the EC2 instance:



2. Assign **AmazonEC2ReadOnlyAccess** privilege to the role.



3. Provide the role name:



Create the security group

You must create a security group for the following ports used by ASE:

- **Port 80:** Accessible by API Clients/ELB
- **Port 443:** Accessible by API Clients/ELB
- **Port 8010:** Accessible by operations to execute CLI commands and REST 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	API clients/ELB
Custom TCP	TCP	80	Same security group
Custom TCP	TCP	443	Same security group
Custom TCP	TCP	8010	Same security group
Custom TCP	TCP	8020	Same security group

Create launch configuration

About this task

Create the launch configuration that the auto-scaling group will use. To create the launch configuration, complete the following steps:

Steps

1. Select the AMI created in *Create an AMI for ASE* section.
2. Create the EC2 instance based on the sizing requirement.
3. Assign the IAM role created in the *Create an IAM Role in the Security, Identity, and Compliance* section to the launch configuration.
4. Complete the creation of launch configuration.

Create an auto-scale group

About this task
Complete the following steps to create the auto scale group:

Steps

1. Create an auto-scale group using the launch configuration created in the previous section.
2. (Optional) Attach the ELB to the auto-scale group created in step 1.
3. Configure the following rules for the auto scale group:
   a. 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.
   b. 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.

Optional: Uninstall the ASE service

If you wish to uninstall the ASE service installed in the *Create an AMI for ASE* section, run the following command:

```
/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 file`s. Note that balancer.log file is not rotated while controller.log file is rotated evevy 24-hours. For more information on ASE logs, see *ASE management, access and audit logs* on page 134

controller.log mesaages:

Log message	Description
unknown cluster uuid	This message is logged in `controller.log` when a ASE node with a different cluster ID or secret key tries to join an ASE cluster. For more information, see *Start ASE cluster* on page 123
resolve error	This message is logged in `controller.log` when ASE is not able to resolve 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.

Log message	Description
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 *ASE license* on page 113.
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. Fore more information, see *Email alerts and reports* on page 137
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 – API JSON configuration file* on page 153
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

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 runnig low on memory because of which ASE drops the client connections.

ABS AI Engine

Introduction

ABS AI Engine is a Java-based distributed system which analyzes API traffic to provide API traffic insight, visibility, and security. API traffic information is received from ASE nodes in log files containing:

- Client details such as device, browser, IP address, and operating system
- Session information including HTTP or WebSocket connections and methods.

These logs are periodically, that is, at every 10 minutes forwarded to ABS nodes for processing. Using machine learning algorithms, ABS generates API traffic insight, anomaly data, and attack insight which identifies clients responsible for attacks. To prevent future attacks, ABS can automatically program inline devices (such as API Security Enforcer) to block clients based on attack lists. PingIntelligence for APIs Dashboard provides visualization of API attack, deception, and metrics.

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
- REST APIs for fetching traffic metrics, anomalies, and attack information
- Email alerts
- Data visualization

Configuring ABS consists of setting up three entities:

1. **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.
2. **ABS AI engine:** One or more ABS instances are configured to receive and process logs and to store results in MongoDB. Ping Identity recommends installing ABS in a cluster for high availability deployments.
3. **PingIntelligence for APIs Dashboard:** The Dashboard uses Elasticsearch and Kibana to render reference graphs for attack types, traffic metrics, and anomaly data. Please refer to ABS Dashboard Admin Guide for installation and configuration information.

Administration

Administering ABS requires understanding:

- Directory structure
- Obfuscating passwords for securing ABS
- Configuring 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

ABS License and timezone

To start 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 total number of transactions subscribed per month and the duration of the license. It is a good practice to *configure your email* before configuring the ABS license. ABS sends an email notification to the configured email ID when the license has expired. Contact the PingIntelligence for APIs sales team for more information. The following points should be noted:
  - **Maximum transaction set to 0**: If your subscription ABS license has zero as maximum transaction, it means that the license has unlimited monthly transaction. Such a license only expires at the end of subscription period.
  - **License expiry**: In case when the subscription license has expired, ABS continues to run until a restart. ABS needs a valid license file to start.

Add an ABS license

If you have not received an ABS license, request a license file from Ping sales. The name of the license file must be `PingIntelligence.lic`. Copy the license file to the`/opt/pingidentity/abs/config` directory and then start ABS.

Update an existing license

If your existing license has expired, obtain a new license from Ping sales and replace the license file in the `/opt/pingidentity/abs/config` directory. Stop and then start ABS after the license file is updated.

Checking the current transaction count

Use the *Admin REST API* on page 309 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
    }
```

ABS timezone

Configure the timezone of ABS host machine to either `local` or in `UTC`. ABS does not provide any configuration to change its timezone. Adjust the timezone by changing the timezone of the host machines' operating system.

Change default settings

It is recommended that you change the default key and password in ABS. Following is a list of commands to change the default values:

Change default JKS password

You can change the default password for KeyStore and the key. Complete the following steps to change the default passwords. Make sure that ABS is stopped before changing the JKS password.

1. **Change the KeyStore password**: Enter the following command to change the KeyStore password. The default KeyStore password is `abs123`.

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

2. **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
```

Start ABS after you have changed the default passwords.

Change abs_master.key

Run the following command to create your own ABS master key to obfuscate keys and password in ABS.

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

Stop ABS: 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
```

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
```

Change admin password

You can change the default admin password by entering 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
```

Change default access and secret key in MongoDB

To change the default access and secret key, stop the ABS nodes and complete the following steps:

1. Connect to MongoDB by entering the following command:

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

`absuser` and `abs123` is the default user name 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>"} )
```

Start the ABS nodes after you have changed the default access and secret key.

Obfuscate passwords

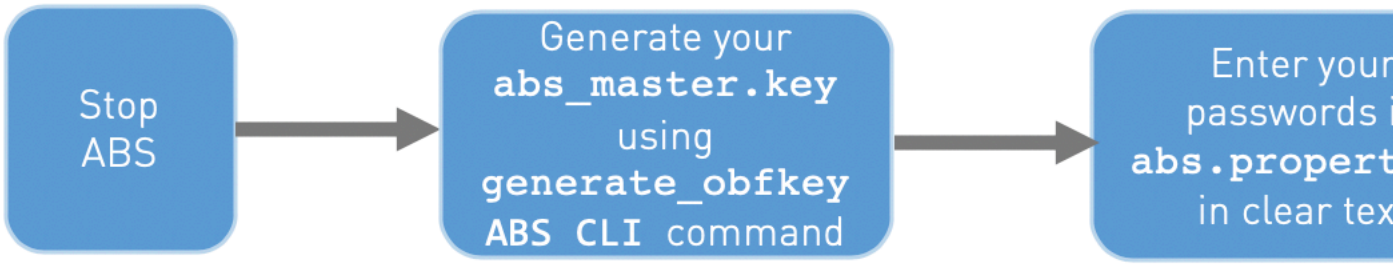
Using ABS command line interface, you can obfuscate the keys and passwords configured in `abs.properties`. The keys and passwords obfuscated include:

- `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`.

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

The following diagram summarizes the obfuscation process:



Generate abs_master.key

You can generate the `abs_master.key` by running 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.

Obfuscate key and passwords

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

```
/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
```

Start ABS after passwords are obfuscated.

ABS POC mode

You can run 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 *REST API attack types* on page 331 .

ⓘ **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.

Configure POC mode

You can install ABS AI engine in POC mode by configuring the parameter during automated installation. For more information on configuring the POC mode at the time of installation, see *Change ABS default settings* on page 59.

If you are using manual installation to install ABS AI engine and MongoDB, configure poc to true in `global_config` of `/pingidentity/abs/mongo/abs_init.js` file. Following is a snippet of `abs_init.js` file:

```
db.global_config.insert({
  "poc": true,
 "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": true,
 "discovery_initial_period" : "1",
 "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"
});
```

Verify the POC mode

Use the ABS Admin REST API to verify whether ABS AI engine is running in the POC mode. The report can be accessed by calling the ABS system at the following URL:

https://<abs_ip>:<abs_port>/v4/abs/admin.

```
{
    "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...
}
```

Start and Stop ABS

Start ABS

To start ABS, run the `start.sh` script located in the `/opt/pingidentity/abs/bin` directory. Change working directory to `/opt/pingidentity/abs/bin`. Then start ABS by typing the following command. To start ABS, you need to accept EULA. You can accept EULA in two ways:

- Scroll through the text on screen and enter `yes` to accept EULA, or
- Use the `--acceptLicense` option with `start.sh` as shown in the screen output below. By using this option, you do not have to scroll through the EULA.

Once the EULA is accepted, ABS creates a `license.accepted` file in the `/opt/pingidentity/abs/config` directory. On subsequent start of ABS, it checks for

```
$ /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
```

To verify whether ABS has started, change the working directory to `data` directory and look for two `.pid` files, `abs.pid` and `stream.pid`. 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 7.1 (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%"
 } ] }
```

Stop ABS

To stop ABS, first stop API Security Enforcer (if it is running) or turn OFF the ABS flag in API Security Enforcer. 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
```

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
```

ⓘ **Note:** Ensure that you stop ABS before performing any of the following tasks:

- When deleting the ABS directory.
- When deleting the data or metadata DB.
- When changing the user permissions.

Omitting to do so will result in excessive logs in the Mongo DB node.

ABS users for API reports

ABS has two type of users to access the API reports and PingIntelligence for APIs 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
  - Cookie, OAuth2 Token, 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"
});
```

ABS directory structure

The directories that ABS creates as part of the installation process 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 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.

`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

ABS supports only TLS 1.1 and TLS 1.2 and requires Open JDK 11.0.2. You can configure SSL by setting the value of `enable_ssl` parameter to true in `pingidentity/abs/mongo/abs_init.js` file. Setting the value to `true` enables SSL communication between ASE and ABS as well as for ABS external REST APIs. Following is a snippet of the `abs.init` file with `enable_ssl` parameter:

```
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"
});
```

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 Passwords* for steps to obfuscate passwords.

If you want to use your own CA-signed certificates, you can import them in ABS.

Import existing CA-signed certificates

You can import your existing CA-signed certificate in ABS. To import the CA-signed certificate, stop ABS if it is already running. Complete the following steps to import the CA-signed certificate:

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>
```

For 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 CA, then append the content to `<your_CA_certificate.crt>` file.

2. Import the certificate and key from the PKCS12 store to Java Keystore by entering the following command. The command requires the destination keystore password. The destination keystore password entered in the command should be same as configured in the *abs.properties* file.

   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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FF
   ```

   Enter the following command:

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

   For 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 `/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
   ```

ABS ports

ABS uses the following ports:

Port number	Description
8080	This port is used by ASE to log in to ABS and also used by Postman to access data to generate API reports
9090	This port is used by ASE to send access logs to ABS
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 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 IP, port, log level, and JKS password	
`host_ip`	The externally visible IP address of the host ABS machine.
`management_port`	Port for ABS to ASE and REST API to ABS communication. The default value is 8080.
`log_port`	Port for ASE to send log files to ABS. The default value is 9090.
`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*.
`log_level`	Log detail captured. The default is INFO. Additional options - DEBUG, ERROR, WARN, FATAL.
ABS performance configurations	
`system_memory`	Memory size in MB allocated to run machine learning jobs. Recommended to be at least 50% of system memory.
`system_json_size`	Memory size in KB allocated for API JSON files. The default is 500 KB.
`runaway_time`	Maximum time in minutes to wait for a job to finish. The default value is 120 minutes.
`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.

`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 configuration	
`mongo_rs`	Comma separated MongoDB replica set nodes IP addresses and port numbers. 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.
`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`.
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.

A sample `abs.properties` file is displayed below.

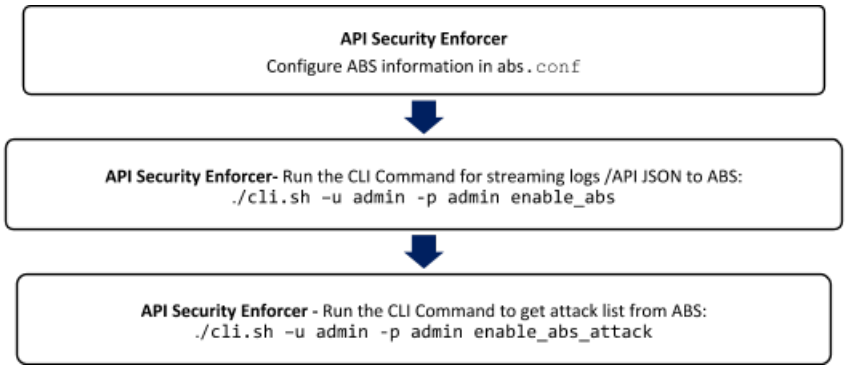
```
# Ping Identity Corporation, ABS config file
# All the keys should be present, leave blank value if not applicable
# ABS node host IP
# If you have multiple network interfaces or if you are running inside a
 Docker, specify the externally visible IP address for ABS to bind
host_ip=127.0.0.1
# REST API port
management_port=8080
# Streaming port
log_port=9090
# Log levels (ALL > DEBUG > INFO > WARN > ERROR > FATAL > OFF)
log_level=DEBUG
# Java KeyStore password
jks_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCul
# MongoDB replica set nodes comma separated IP addresses and port numbers.
 For example, <IP1>:<Port>, <IP2>:<Port>, <IP3>:<Port>. Maximum three nodes
 can be configured.
mongo_rs=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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FF
# 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
# Time to mark a job runaway in minutes
runaway_time=120
# 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
# Memory for ASE JSON (unit is KB)
system_json_size=8192
# 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
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
```

Connect ABS to API Security Enforcer

Before connecting ABS, API Security Enforcer must be installed. For more information on installing and configuring API Security Enforcer, see the *ASE Admin guide*.

The following diagram summarizes the process of connecting ABS to API Security Enforcer:



The following is a sample `abs.conf` file which is part of the API Security Enforcer (ASE):

```
; 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 API Security Enforcer cluster to ABS, configure the `abs.conf` file on any API Security Enforcer in the cluster and run the CLI commands. This ensures all the API Security Enforcer nodes in the cluster will be updated to connect with ABS.

If ABS is running in cluster mode, choose the IP address and port from any ABS node to add to the `abs.conf` file in API Security Enforcer.

Dataflow

API Security Enforcer connects 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 API Security Enforcer.

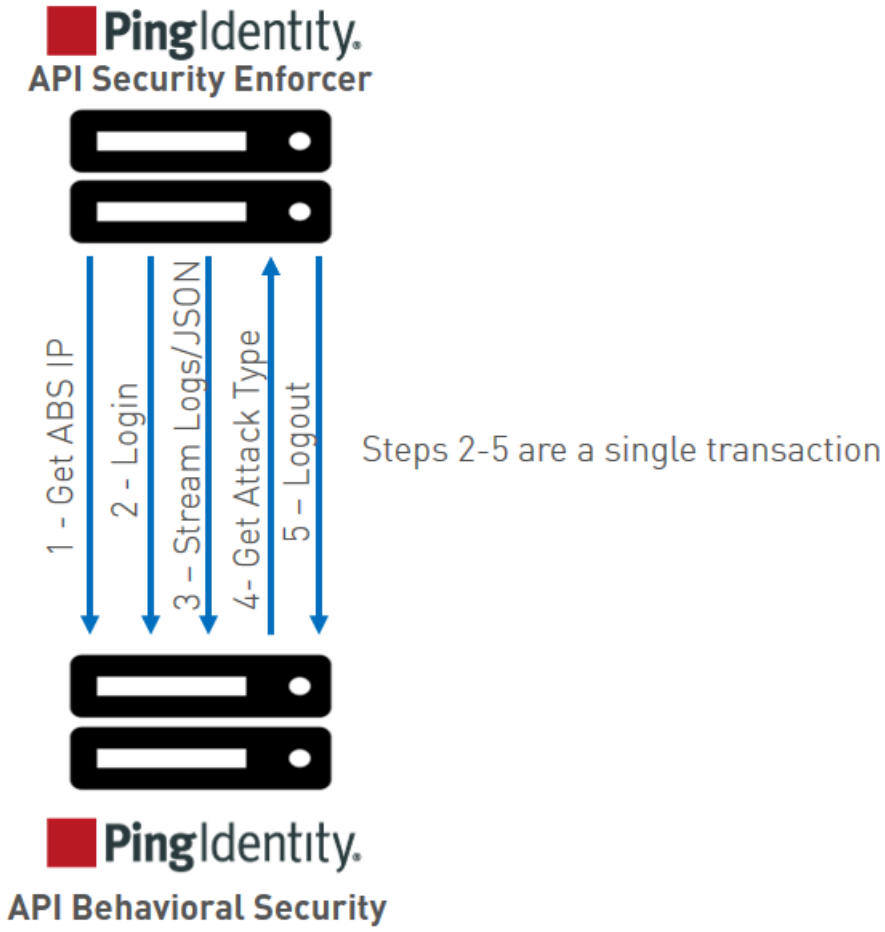
After API Security Enforcer receives the IP address, it establishes a session with ABS by sending the secret and access keys (step 2). After successful authentication, API Security Enforcer streams the access log files and API JSON files to the ABS node (step 3). After sending the files, it receives the attack lists (only available if blocking is activated for API Security Enforcer) from ABS (step 4). When the transaction is complete, API Security Enforcer logs out from ABS (step 5).

ABS uses machine learning (ML) algorithms to discover attacks, anomalies, and other traffic information. It stores incoming API Security Enforcer 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 API Security Enforcer – ABS Dataflow.

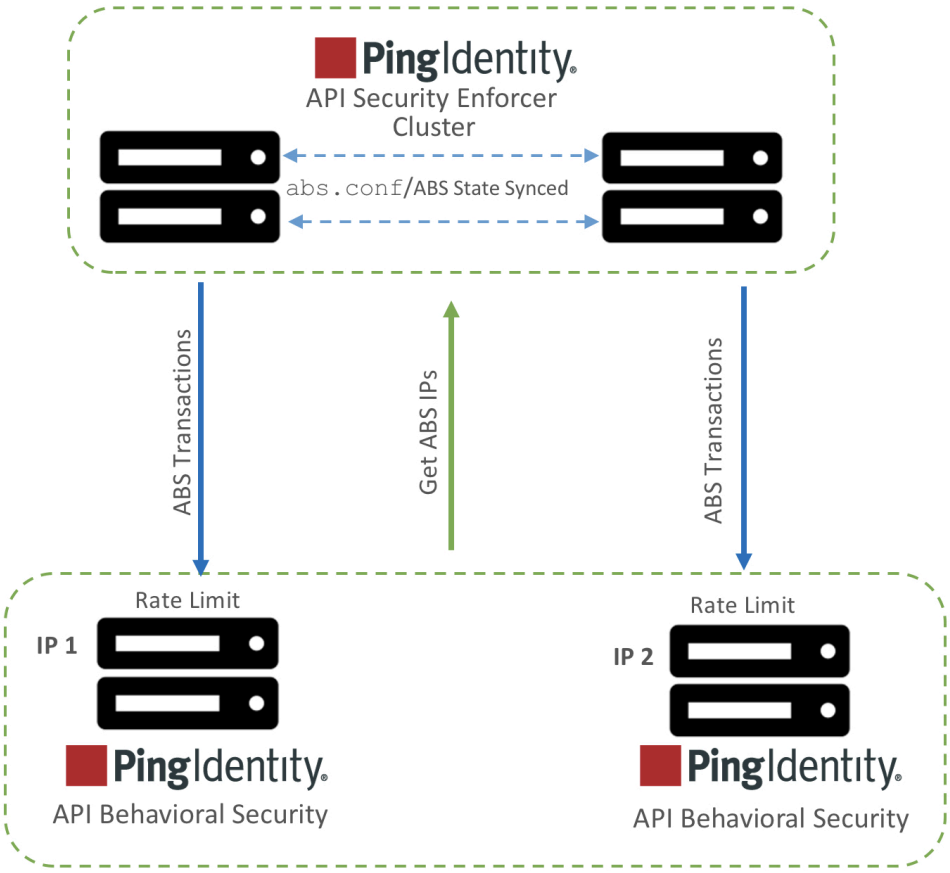
Stand-alone mode

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.



Cluster mode

In cluster mode, API Security Enforcer 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.

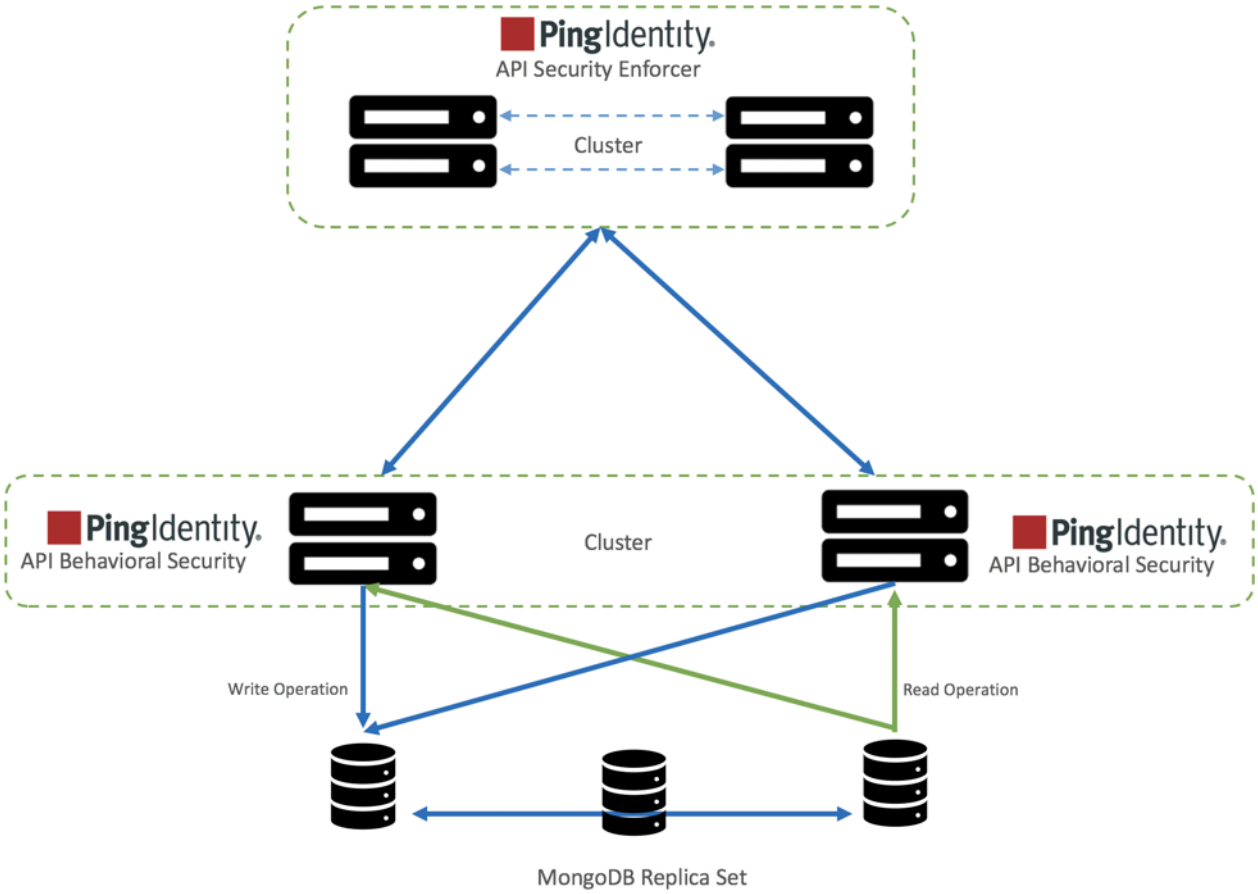


ABS cluster

An 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
- 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:



To configure an ABS cluster, complete the following steps:

1. *Install MongoDB in a replica set*
2. *Connect ABS to MongoDB*

To set up an ABS cluster, no separate steps have to be completed. To create an ABS cluster, add an ABS node and connect it to MongoDB primary node. Since ABS forms a stateless cluster, the information of all the nodes in the cluster is fetched by ABS nodes from MongoDB.

Scale down ABS cluster: To scale down the cluster, *stop* the ABS node that you wish to remove from the cluster. Edit the `abs.properties` file to remove MongoDB IP address.

ABS logs

The active 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 which 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 ABS command line. For example:

```
/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

Purge MongoDB data

The ABS MongoDB purge script dumps and/or deletes processed ABS 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 `/opt/pingidentity/abs/util` directory. Copy the script from the `util` directory to your MongoDB primary node.

The script offers three options:

1. Dump data into a directory and then purge it
2. Only dump data
3. Only purge data

To execute the script, enter the following information on the command line:

- **MongoDB credentials:** `mongo_username, mongo_password in abs.properties`
- **Database name and port number:** `data_dbname, mldata_dbname, and mongo_master_port in abs.properties`
- **Days of data to retain:** minimum of one and maximum of 365 days
- The path to dump the data
- If your MongoDB installation is configured to use TLS/SSL, use the --tls option. The following examples assumes that MongoDB is configured to use TLS/SSL.

For more information on the purge script parameters, run the purge help script from the MongoDB command line:

```
/opt/pingidentity/mongo/purge_mongo.sh –help
```

By default, the script dumps all data and then removes processed data older than seven days. Here are examples of the three options:

1. **Dump and purge example:** The following example shows both `abs_data` and `abs_mldata`

   ```
   /opt/pingidentity/mongo/purge_mongo.sh -u absuser -p abs123 --tls --
   data_db abs_data --mldata_db abs_mldata --auth_db admin --port 27017 -d 80
    -l /tmp
   ```

   Dumps all log files to `/tmp` and purges log files greater than 80 days old.
2. **Dump example:**

   ```
   /opt/pingidentity/mongo/purge_mongo.sh -u absuser -p abs123 --tls --
   data_db abs_data --auth_db admin --port 27017 -d 45 -l /tmp --dump_only
   ```

   Dumps all log files to `/tmp` and purges log files greater than 45 days old.

   The following example shows dumping only the ABS data:

   ```
   /opt/pingidentity/mongo/purge_mongo.sh -u absuser -p abs123 --tls --
   data_db abs_data --auth_db admin --port 27017 -d 45 -l /tmp --dump_only
   ```

   Dumps all log files to `/tmp` and purges log files greater than 45 days old.
3. **Purge example:**

   ```
   /opt/pingidentity/mongo/purge_mongo.sh -u absuser -p abs123 --tls --
   data_db abs_data --auth_db admin --port 27017 -d 80 --purge_only
   ```

   Purges log files greater than 80 days old.

   ⓘ **CAUTION:** Once the MongoDB data is purged, it cannot be retrieved.

   The following example shows purging only the `mldata`:

   ```
   /opt/pingidentity/mongo/purge_mongo.sh -u absuser -p abs123 --mldata_db
    abs_mldata --auth_db admin --port 27017 -d 80 --purge_only
   ```

Reset MongoDB

ABS AI engine provides a script to factory reset MongoDB data. 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.

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 SSL, use the --ssl option. The following examples assume that MongoDB is configured to use TLS.

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
```

Reset ABS and machine learning data: 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
```

Reset only machine learning (ml) data: 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
```

Reset only ABS data: 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
```

The following snippet shows the output when the reset MongoDB script is run:

```
./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]
```

The reset script does not delete the following meta data:

- ABS cluster information
- ABS configuration
- Global configuration from `abs_init.js` file
- Scale configuration from `abs_init.js` file
- Dictionary generated by ABS AI engine

Verifying MongoDB reset script: To verify that the MongoDB reset script executed successfully, run the ABS Admin REST API. The output should not show any 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. Following is a sample output of Admin API after MongoDB reset script is run:

```
{
    "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
            },
            {
                "id": "cookie",
                "ttl": 0
            },
            {
                "id": "access_token",
                "ttl": 0
            },
            {
                "id": "api_key",
                "ttl": 0
            },
            {
                "id": "username",
                "ttl": 0
            }
        ]
    }
}
```

Add a new member to existing MongoDB replica set

This topic discusses the steps to add a new node to an existing MongoDB replica set.

Prerequisites:

- An active replica set.
- A new MongoDB system accessible by the replica set.

> ⓘ **Note:** absrs01 is the name of the replica set used in the following steps.

Complete the following steps to add a node to an existing replica set:

1. Create the MongoDB directory structure: create mongo, data, logs, and key directory 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 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
```

3. Update 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 in to the same location `/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})
```

> ⓘ **Note:** On executing step-six, the state of the new node will change to STARTUP2. This indicates that the synchronization between the replica set and the new node has started.

7. Verify if the new node is added as a Secondary node to the replica set using the following command.

```
absrs01:PRIMARY> rs.status()
```

Related links

- *https://docs.mongodb.com/manual/tutorial/expand-replica-set/*

Remove a member from MongoDB replica set

This topic discusses the steps to remove a node from an existing MongoDB replica set.

To remove a node from an existing replica set:

1. Connect to the node that you wish to remove and shut down the MongoDB on it using the following command.

```
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")
```

ⓘ **Note:** absrs01 is the name of the replica set used in the following steps.

Related links

https://docs.mongodb.com/manual/tutorial/remove-replica-set-member/

Email alerts and reports

ABS sends e-mail notifications under two categories:

- **Alerts** – event-based updates to notify administrators of potential issues
- **Reports** – standard reports sent every 24 hours at 00:00:00 hours 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.

```
#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

Threshold values are configured in the `/opt/pingidentity/mongo/abs_init.js` file which is in the `mongo` directory. An email alert is sent based on the following category of events. These events are also logged in the `abs.log` file.

- **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.

- **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.

Here is a snippet of an `/opt/pingidentiy/mongo/abs_init.js` file for email alerts on the MongoDB node. You can configure any of these values as per your requirement. It is a good practice to set the values of email alerts before *configuring MongoDB* and the `abs_init.js` file. `scale_up` is for the upper threshold, while `scale_down` is for the lower threshold. If you want to change the threshold values after the system is running, then you have to manually change the values in MongoDB and restart the ABS node.

```
db.scale_config.insert({
 "scale_up": [{
  "resource": "memory",
  "threshold": "70%",
  "monitor_interval": "30minutes"
 }, {
  "resource": "cpu",
  "threshold": "70%",
  "monitor_interval": "30minutes"
 }, {
  "resource": "disk",
  "threshold": "70%",
  "monitor_interval": "30minutes"
 }],
 "scale_down": [{
  "resource": "memory",
  "threshold": "10%",
  "monitor_interval": "300minutes"
 }, {
  "resource": "cpu",
  "threshold": "10%",
  "monitor_interval": "300minutes"
 }, {
  "resource": "disk",
  "threshold": "10%",
  "monitor_interval": "300minutes"
 }]
});
```

Following is a template for alerts:

```
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 PingIntelligence 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.
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

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
- **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.
```

ABS REST API format

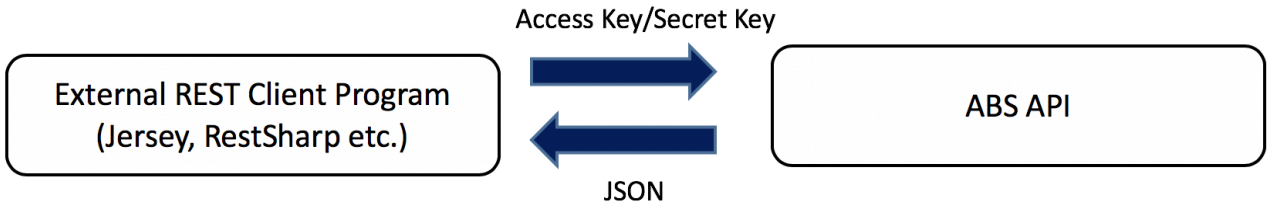
ABS provides external REST APIs which are used to access JSON reports providing deep insight into the following:

- 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)
- 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 include:

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 10th April, 6 PM to 14th April, 3 PM, the `earlier_date` would be 10th April, 6 PM.
`later_date`	The time to check the results back in time. For example, to check results from 10th April, 6 PM to 14th April, 3 PM, the `later_date` would be 14th April, 6 PM.

The following `access_key` and `secret_key` are the keys that were defined in the `abs_init.js` file. Note that ":" (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.

Admin REST API

The Admin REST API reports on ABS cluster node resources including IP address, operating system, CPU, memory, and filesystem usage. It also reports MongoDB node information including IP address, node type, and status. Finally, it provides status on attack detection and reporting on APIs.

The report can be accessed by calling the ABS system at the following URL:

`https://<ip>:<port>/v4/abs/admin`

The following is a sample JSON report.


```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, a
    "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": ""
            }
        }
    ],
    "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",
                "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
            }
        ]
    }
}
```

Percentage disk usage limit: The percentage disk usage limit is configured in the `/pingidentity/abs/mongo/abs.init.js` file. It is a good practice to configure this value before initializing MongoDB and ABS. ABS stops accepting access log files from ASE when the configured `percentage_diskusage_limit` is reached. An *email alert* is sent to the configured email ID and also logged in the `abs.log` file.

You can update the disk usage limit using the `updates.sh` script available in the `/opt/pingidentity/abs/util`. 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.

Access script help by logging into the MongoDB primary machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```

Following is an example of the script:

```
./update.sh -u absuser -p abs123 --db abs_metadata --auth_db admin --port
 27017 --percentage_diskusage_limit 80
updating percentage_diskusage_limit to 80
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
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=24
url_limit=100
response_size=100
window_length=24
discovery_subpath=3
percentage_diskusage_limit=80
```

Configure TTL for client identifiers

Admin API with PUT method is used to configure the length of time to maintain blacklist entries for the different client identifiers, for example, IP address, token, cookie, and API key. For more information on configuring TTLs, see *TTL for client identifiers*

AI Engine training

The ABS AI Engine needs to be trained before it can detect attacks on API services. The AI engine training is governed by global variables which are configured in the `/opt/pingidentity/abs/mongo/`

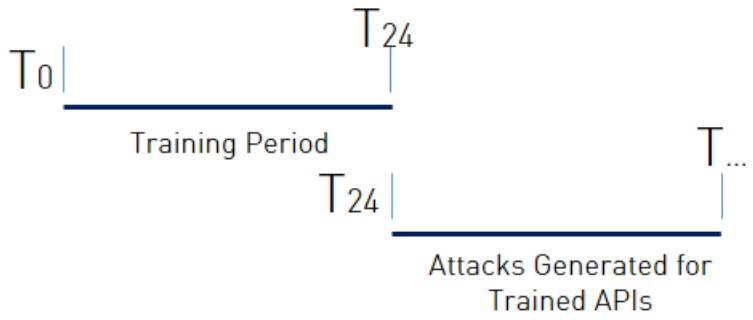
`abs_init.js` file. The AI training runs for the minimum training time set in the `abs_init.js` file but a minimum amount of data must also be received before the training period is complete for a given API. You can check the *training status* by using the *ABS Admin REST API.*.

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

ABS AI engine can be trained in a live environment by analyzing ASE access logs to build its model. When ABS first receives traffic for a new 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.



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 in the `abs_init.js` file. 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. 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.
`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 APIs. The default and maximum value for `window_length` is 24-hours. The training period should be longer than the `window_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 ABS decides that the session has terminated. Default value is 30-minutes. You can configure it to any value in minutes. ⓘ **Note:** This variable only applies to account take over attack.

Following is a snippet from the `abs_init.js`file showing the variables:

```
db.global_config.insert({
  "poc": false,
 "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": true,
 "discovery_initial_period" : "1",
 "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"
});
```

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	When true, SSL communication is enabled between ASE and ABS, and for external systems making rest API calls to ABS. See *Configure SSL* on page 288on page 10 for more information.

Training period status

ABS training status is checked using the ABS Admin API which returns the training duration and prediction mode. 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. The API 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
```

ⓘ **Note:** ABS only detects attacks after the training period is over. During training, no attacks are detected.

Update the training variables

ABS provides an update.sh script to update the training related variables in the global configuration of abs_init.js file. Using the script, you can update the following variables:

- 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 update.sh script available in the util directory. Review the following use cases before changing the training and threshold period. In all the use cases, the default training period is assumed to be 24-hours. You can update the default values before starting the system by editing and saving the values in the abs_init.js file.

ⓘ **CAUTION:** If you want to extend the training period, it is a best practice to add new APIs after the training period is adjusted to avoid APIs completing a shorter training period.

You can also use Global Configuration REST API to update the training variables. For more information see, *Global configuration update REST API* on page 329

Update the training interval

Increase the training period

You can increase the training period by executing the update script.

Case 1 – The API model is under training, that is, the training period is not over.

System Behavior – 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.

Case 2 – The API model has completed the training process.

System Behavior – 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 executing the update script.

Case 1 – The API model is in the training process but has not reached the duration of the new training period.

System Behavior – 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.

Case 2 – The API model is in the training process and the new training duration is less than the current AI model trained duration.

System Behavior – 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.

Case 3 – API model has completed the training process.

System Behavior – Decreasing the training period has no effect on trained APIs. Any new APIs will use the new training period.

Execute 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 node. The training period and threshold can be changed simultaneously or individually.

ⓘ **Note:** After executing the script, stop and start all ABS nodes for the updated values to take effect.

Access script help by logging into the MongoDB primary machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```

Example Change the training period to 48 hours

```
/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
```

Tune thresholds for false positives

ABS automatically generates attack thresholds which 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 start-up by modifying `attack_update_interval` in the `abs_init.js` file or anytime by using the `update.sh` script available in the `util` directory. The minimum value is 1-hour as sufficient traffic is required to update the model.

Following is a snippet of `abs_init.js` file:

```
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": true,
 "discovery_initial_period" : "1",
 "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"
});
```

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 logging into the MongoDB machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```

Example: change the training period and threshold interval together

```
/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
```

Check 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) is 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.

Change attack thresholds

Ping Identity recommends using the automatically generated system thresholds in your production operations. However, if attacks are detected for legitimate traffic (i.e. 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 variable(s) 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 value(s) 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, the normal threshold (tn), the extreme threshold (tx), or either threshold can be individually set.

You can also use *Attack management* on page 19 in Dashboard to tune threshold values for a specific client identifier. For more information, see *Tune thresholds.*

> ⓘ **Note:** Make sure that you are in *user* mode before changing the threshold manually.

Change 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`. Here is an example:

Values at end of training period:

- `Tn = 12`
- `Tx = 16`
- `Gap = 4 (Tx-Tn)`

Threshold API is used to set `Tn=13` for an API variable.

- `Tx = 17` (Gap value of 4 is automatically added to new `Tn`value)

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.

Change threshold value Tn and Tx together

Both Tn and Tx can be changed 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*.

How to configure threshold value

To manually set a threshold, use the PUT method with the following ABS attack API:

https://<ip_address>:<port>/v4/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).

Manually set thresholds

The threshold API with PUT method sets the operation mode for the variable by configuring mode to system or user. The following snippet of Threshold API with 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 wish 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 value of tn and tx both are 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 API response is displayed below:

```
{
 "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 API response is displayed below:

```
{
 "message": success: "Thresholds set to system mode for given variables.",
 "date": "Mon Jan 06 15:36:05 IST 2018"
}
```

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* on page 431 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 *Enable or disable attack IDs* on page 345.

API discovery and configuration

The ABS AI Engine works in tandem with ASE to automatically discover new and unknown APIs 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 PingIntelligence for APIs Dashboard. For more information, see *Discovered APIs* on page 471.

Following is the summary of the steps to configure API discovery in your environment:

1. Enable ABS in ASE
2. Define `root` API JSON in ASE. 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 in `abs_init.js` file.

   When MongoDB is installed, the `abs_init.js` file is copied to MongoDB. Use the `update.sh` script to edit the default values related to API discovery. For more information on update script, see *Manage discovery intervals* on page 328.

Configuration in ASE for API discovery

- **Enable ABS in ASE** Enable ABS 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:

```
./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
```

- **Configure root API in ASE**: ABS discovers APIs in your environment only when `root` API is defined in ASE. 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 "`*`". 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": ""
        },
```

  A sample `root` API ships with ASE in `/pingidentity/ase/config/api` directory.

> ⓘ **Note:** 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`.

- **API JSON configuration (Optional)**: You can optionally configure the settings for `cookie`, `oauth2_access_token`, `apikey_qs`, or `apikey_header` in the `root` API JSON file in ASE.

  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.

ABS configuration for API discovery: Configure API discovery in ABS by setting the `api_discovery` parameter to true in `abs_init.js` files.

```
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": true,
 "discovery_initial_period" : "1",
 "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"
```

```
});
```

The following table summarizes the variables related to API discovery that you need to configure in `abs_init.js` file. If you want update the values on an already running system, use the `update.sh` script. For more information on update script, see *Manage discovery intervals* on page 328:

API discovery variables

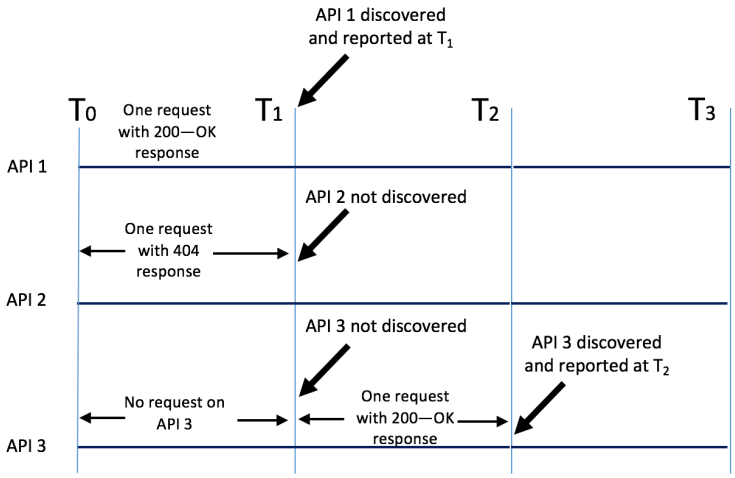
Variable	Description
`api_discovery`	Set this variable to `true` to switch on API discovery. To switch off API discovery, set it to `false`. The default value is `true`.
`discovery_initial_period`	The initial time in hours during which APIs are discovered in your API ecosystem. The default and minimum 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 subpaths that is discovered in an API. The minimum value is 1 and maximum value is 6. For more information, see *Discovery Subpaths* on page 325.
`url_limit`	Defines the maximum number of URLs that are reported in a discovered API.

API discovery process

ABS discovery process starts when ASE sends the access log files to ABS. The discovery process and reporting interval are defined by the variables configured in `abs_init.js` file, as explained in the *API discovery and configuration* on page 321 topic.

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* 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:



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.
`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* on page 471 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.

Discovery Subpaths

Before starting API discovery, it is important to configure the subpath depth which allows the AI Engine to accurately detect the API environment. Subpath depth provides the number of sub-paths for a unique API definition. Here are examples of `discovery_subpath` values:

- **"1",** example: `/atmapp` is basepath for `/atmapp/zipcode, /atmapp/update`, etc.
- **"2",** example: `v1/atmapp` is basepath for `v1/atmapp/zipcode, v1/atmapp/update`, etc.
- **"3",** example: `v1/cust1/atmapp` is basepath for `v1/cust1/atmapp/zipcode`, etc.

The `discovery_subpath` parameter is configured in the `abs_init.js` file and defines the number of sub-paths in the basepath of the API. The default value is set to 1 and the maximum value is 6. The `url_limit` parameter defines the maximum number of subpaths in a discovered API. The default value is 100.

```
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" : "1",
 "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"
});
```

Updating url_limit and discovery_subpath: 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.

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
```

ABS Discovery API

The Discovery API uses the GET method to display the discovered API details and is reported only when the `host`, `basepath`, `schemes`, `paths`, and `responses` information is populated. ABS provides the following external REST API which uses the GET method to view the discovered APIs:

URL: `/v4/abs/discovery`

Following is a snippet of the summary output of `discovery` API:


```
{
    "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",
            "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:


```
{
    "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"
    ],
    "jwt": {
        "username": "username",
        "clientid": "client_id",
        "location": "h:authorization:bearer"
    },
    "paths": {
        "/shop-books3": {
            "GET": {
                "produces": [
```

```
                    "text/html"
                ],
                "responses": {
                    "200": {
                        "description": "OK"
                    }
                }
            }
        }
    }
}
```

ⓘ **Note:** If ASE is deployed in sideband mode, then backend host field in the output shows the 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.

Manage discovery intervals

You can enable or disable discovery and also 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 APIs are reported 1-hour from the time when ASE sends access logs. You can update these default values using the update script.

Execute 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. You can change the training period and threshold simultaneously or individually.

You can access the script help by logging in to the MongoDB primary machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```

Example:

```
/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
```

Global configuration update REST API

ABS provides a REST API to update global configurations related to AI engine training and API discovery. These global configurations are part of `pingidentity/abs/mongo/abs_init.js` file. The updated global configuration values take effect immediately. Following is the list of six global configurations that you can update using the `globalconfig` API:

- attack_initial_training
- attack_update_interval
- api_discovery
- discovery_initial_period
- discovery_subpath
- discovery_update_interval

ⓘ **Note:** If you want to update the other global configurations, see the update.sh script available in the util directory. For more information on update.sh script see, *Update the training variables* on page 315 and *Manage discovery intervals* on page 328.

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* on page 286.

URL - `https//<abs_host>:<abs_port>/v4/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.

```
{
    "company": "ping identity",
    "name": "api_globalconfig",
    "description": "This report contains status information of ABS global
 configurations",
    "global_config": {
    "attack_initial_training": 24,
    "attack_update_interval": 24,
    "api_discovery": true,
    "discovery_initial_period": 1,
    "discovery_subpath": 1,
    "discovery_update_interval": 1
    }
}
```

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_subpath": 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.

REST API attacks

For each API, the API JSON file (see *API Security Enforcer Admin Guide* for information) determines whether the attacks and other reports are based on OAuth token, API Keys, username, cookie, or IP address. An environment with multiple APIs can support a mixture of identifier types in a single ABS AI Engine. Client identifier examples include. Client identifier examples include:

- **API using OAuth2 tokens –** When an API JSON file is configured with OAuth2 token `parameter = true`, then attack information is associated with the OAuth2 access token used by the hacker. Configuring the OAuth2 token parameter is recommended when access tokens are present as it is a unique client identifier that eliminates issues identified below with IP addresses.
- **APIs using API Keys**:When API JSON file is configured with API Key either in the query string or the header, ABS detects attacks on the value of the API Key. 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 blacklist 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 attacks 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.
- **API JSON without a cookie or token parameter –** When cookie and OAuth2 token parameters are not configured, all 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 unless cookies or tokens are used.

REST API attack based on username:

In some sideband deployments, for example, PingAccess with PingFederate, the username of the accessing client is available via RFC 7662 token introspection or other techniques. ABS AI Engine detects attacks based on the username. Unlike other client identifiers, username is not configured in the API JSON file. The ABS AI engine detects the username from the metadata sent by ASE.

To change the client identifier for an existing API, save the API JSON with a new name and update the configuration to include the new client identifier parameter. ABS then re-trains the model for this API and starts detecting attacks.

REST API attack types

ABS AI Engine reports on REST API attacks by delivering reports on per API attacks, that is, client attack targeted a single API. ABS AI engine also reports across API attacks, that is, client attack targeted multiple APIs.

Per API attacks: These attacks are reported on a specific API and is based on activity from a client using an OAuth token, cookie or an IP address. Each attack type is assigned a type ID and can be accessed using the `attack` REST API of ABS. Entering type ID 0 reports on all attacks on the specified API except for attack types which are analyzed across APIs.

Use the following ABS REST API to access different attack types: `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>`.

For example, `https://192.168.11.166:8080/v4/abs/attack?later_date=2019-12-31T18:00&later_date=2019-10-25T13:30&api=shop&type=1`

The following table lists the attack types for individual APIs:

Per API attacks

Attack Type	Type ID
Data Exfiltration Attack Type 1	1
Single Client Login Attack Type 1	2
Multi-Client Login Attack	3
Stolen Token Attack Type 1 (Token)	4
Stolen Cookie Attack Type 1 (Cookie)	4
API Memory Attack Type 1	5
API Memory Attack Type 2	6
Cookie DoS Attack	7
API Probing Replay Attack Type 1	8
API DDoS Attack Type 1	9
Extreme Client Activity Attack	10

Extreme App Activity Attack	11
API DoS Attack	12
API DDoS Attack Type 2	13
Data Deletion Attack	14
Data Poisoning Attack	15
Data Exfiltration Attack Type 2	21
Content Scraping Type 2	28
Unauthorized Client Attack	29
Header Manipulation Attack	37

Across API attacks:

These attacks are detected across APIs and are based on activity from a client username or client using an OAuth token, cookie or an IP address. For example, a hacker with a token may execute attacks which span across multiple APIs.

Use the following ABS REST API to access different attack types: `https://<ABS_IP:port>/v4/abs/attack?later_date=yyyy-mm-ddThh:mm&later_date=yyyy-mm-ddThh:mm&type=<type_id>`.

For example, `https://192.168.11.166:8080>/v4/abs/attack?later_date=2019-12-31T18:00&later_date=2019-10-25T13:30&type=18`

The following table lists the attack types for individual APIs:

Across API attacks

Attack Type	**Type ID**
Stolen Token Attack Type 2	16
Stolen Cookie Attack Type 2	17
API Probing Replay Attack Type 2 (Cookie)	18
API Probing Replay Attack Type 2 (Token)	19
API Probing Replay Attack Type 2 (IP)	20
Excessive Client Connections (Cookie) ⓘ **Note:** Applicable only for Inline ASE deployment. For more information, see **Excessive Client Connections** section below.	22
Excessive Client Connections (Token) ⓘ **Note:** Applicable only for Inline ASE deployment. For more information, see **Excessive Client Connections** section below.	23

Excessive Client Connections (IP) ⓘ **Note:** Applicable only for Inline ASE deployment. For more information, see **Excessive Client Connections** section below.	24
Content Scraping Type 1 (Cookie)	25
Content Scraping Type 1 (Token)	26
Content Scraping Type 1 (IP)	27
Single Client Login Attack Type 2	30
Stolen API Key Attack	31
API Probing Replay Attack Type 1	32
API Probing Replay Attack Type 2	33
API Probing Replay Attack Type 1	34
API Probing Replay Attack Type 2	35
Sequence Attack	36
Abnormal API Access	38

Excessive Client Connections

Excessive client connections attack has three attack IDs, 22, 23, and 24 for IP, cookie, and token. These three attack IDs are disabled by default when you install PingIntelligence. However, you can enable these attacks for PingIntelligence inline deployment by using the `attackstatus` REST API. For more information, see *Enable or disable attack IDs*. Attack IDs 22,23, and 24 are not available for PingIntelligence sideband deployment since ASE does not receive the API traffic directly from the client.

For more information on Inline and Sideband ASE deployment modes, see *ASE deployment modes*.

Attacks based on username activity

ABS AI Engine detects attacks based on behavior from the username accessing API services. To capture the username information, PingIntelligence must be deployed in sideband mode with an API gateway that supports capturing username information. User information can also be captured if the incoming request has a JSON Web Token (JWT).

If the incoming request has JWT, then username attacks can be detected for both inline and sideband deployment. For more information on ASE support for JWT, see *Extract user information from JWT in inline mode* on page 207 and *Extract user information from JWT in sideband mode* on page 161. Following is a list of PingIntelligence and API gateways integration that support capturing username information:

- *Akana API gateway sideband integration* on page 489
- *Axway sideband integration* on page 549
- *PingIntelligence Apigee Integration* on page 513
- *Mulesoft sideband integration* on page 616
- *PingAccess sideband integration* on page 665
- *NGINX sideband integration* on page 631
- *NGINX Plus sideband integration* on page 646
- *PingIntelligence WSO2 integration* on page 677

Note the following points for ABS AI engine to detect username based attacks:

- OAuth token parameter, oauth2_access_token, must be configured in API JSON in ASE. For more information on API JSON definition see, *Defining an API – API JSON configuration file* on page 153
- The incoming request must have an OAuth token in it for ABS AI engine to detect username based attacks

Detected attacks based on username

Attack Type	Description	id	Single or Across APIs
API Probing Replay Attack Type 1	Probing or breach attempts on an API service – also called fuzzing - Username	34	Across APIs
API Probing Replay Attack Type 2	Probing an API service over an extended time period - Username	35	Across APIs
Sequence Attack	Abnormal sequence of API transactions	36	Across APIs
Abnormal API Access	Abnormal user behavior when accessing API services	38	Across APIs

ⓘ **Important:** While reporting an abnormal sequence, if username is available with the API ecosystem, ABS reports username or else it reports OAuth token.

Attacks based on API Key activity

ABS AI Engine detects attacks based on client activity using an API Key. The following table lists the attacks detected on a single API or across multiple APIs.

Detected attacks based on API Keys

Attack Type	Description	id	Single or Across APIs
Stolen API Key Attack	A stolen API Key is being used to attack an API service.	31	Across APIs
API Probing Replay Attack Type 1	Probing or breach attempts on an API service – also called fuzzing - API Key	32	Across APIs
API Probing Replay Attack Type 2	Probing an API service over an extended time period - API Key	33	Across APIs

Attacks based on cookie activity

ABS AI Engine detects attack based on client activity using a Cookie. The following table lists the attacks detected on a single API or across multiple APIs.

Detected attacks based on cookie activity

Attack Type	Description	id	Single or Across APIs
Data Exfiltration Attack Type 1	Data is being extracted via a REST API service.	1	Single API
Stolen Cookie Attack	A stolen cookie is being used to attack an API service.	4	Single API
API Memory Attack Type 1	Flooding of an API service with data or code.	5	Single API
API Memory Attack Type 2		6	Single API
Cookie DoS Attack	Client attacking session management service with a high volume of cookies.	7	Single API
API Probing Replay Attack	Probing or breach attempts on an API service – also called fuzzing.	8	Single API
API DDoS Attack Type 1	A DDoS or distributed attack is disrupting an API service.	9	Single API
Data Deletion	Excessive data deletion activity on an API service.	14	Single API
Data Poisoning	Extreme create or update activity received on an API service.	15	Single API
Stolen Cookie Attack Type 2	A stolen cookie is being used to attack an API service.	17	Across APIs
API Probing Replay Attack Type 2	Probing an API service over an extended time period - Cookie	18	Across APIs
Data Exfiltration Attack Type 2	Data is being extracted via a REST API service over an extended time period.	21	Single API

Excessive Client Connections ⓘ **Note:** The **Excessive Client Connections** attack type is disabled by default. For more information, see *REST API attack types* on page 331.	Client is establishing an excessive number of TCP connections.	22	Across APIs
Content Scraping Type 1	Client abnormally accessing API content	25	Across APIs
Content Scraping Type 2	Client abnormally accessing API content over an extended time period	28	Single API
Header Manipulation	Probing an API using malicious headers	37	Single API

.

Attacks based on token activity

ABS AI Engine detects attacks based on client activity using an OAuth Token. The following table lists the detected attacks on a single API or across multiple APIs

Attack Type	**Description**	`type_id`	**Single or Across APIs**
Data Exfiltration Attack Type 1	Data is being extracted via a REST API service.	1	Single API
Stolen Access Token Attack	A stolen access token is being used to attack an API service.	4	Single API
API Memory Attack Type 1	Flooding of an API service with data or code.	5	Single API
API Memory Attack Type 2		6	Single API
API Probing Replay Attack	Probing or breach attempts on an API service – also called fuzzing.	8	Single API
API DDoS Attack Type 1	A DDoS or distributed attack is disrupting an API service.	9	Single API
Data Deletion	Excessive data deletion activity on an API service.	14	Single API
Data Poisoning	Extreme create or update activity received on an API service.	15	Single API

Stolen Token Attack Type 2	A stolen token is being used to attack an API service.	16	Across API
API Probing Replay Type 2	robing an API service over an extended time period - Token	19	Across APIs
Data Exfiltration Attack Type 2	Data is being extracted via a REST API service over an extended time period.	21	Single API
Excessive Client Connections ⓘ **Note:** The **Excessive Client Connections** attack type is disabled by default. For more information, see *REST API attack types* on page 331.	Client is establishing an excessive number of TCP connections.	23	Across APIs
Content Scraping Type 1	Client abnormally accessing API content	26	Across APIs
Content Scraping Type 2	Client abnormally accessing API content over an extended time period	28	Single API
Sequence Attack	Abnormal sequence of transactions	36	Across APIs
Header Manipulation	Probing an API using malicious headers	37	Single API

ⓘ **Important:** ABS also reports Sequence attack on OAuth token. However, if a username is available, it is first reported against username.

Attacks based on IP activity

The following table lists the REST API attacks detected using an IP address as the client identifier. The attacks can be on a single API or across APIs

Attack Type	Description	id	Single or Across APIs
Data Exfiltration Attack	Data is being extracted via a REST API service.	1	Single API
Single Client Login Attack Type 1	Login service attacked by a bot or rogue client.	2	Single API
Multi-Client Login Attack	Login service is under DDoS attack by bots.	3	Single API

API Memory Attack Type 1	Flooding of an API service with data or code.	5	Single API
API Memory Attack Type 2		6	Single API
API Probing Replay Attack	Probing or breach attempts on an API service – also called fuzzing.	8	Single API
API DDoS Attack Type 1	A DDoS or distributed attack is disrupting an API service.	9	Single API
API DoS Attack	Client (IP) sending high volumes of requests to overload application services	12	Single API
API DDoS Attack Type 2	Multiple clients (IP botnet) sending high volume traffic to overload the API service	13	Single API
Data Deletion	Excessive data deletion activity on an API service.	14	Single API
Data Poisoning	Extreme create or update activity received on an API service.	15	Single API
API Probing Replay Type 2	Probing an API service over an extended time period - IP	20	Across APIs
Data Exfiltration Attack Type 2	Data is being extracted via a REST API service over an extended time period.	21	Single API
Excessive Client Connections ⓘ **Note:** The **Excessive Client Connections** attack type is disabled by default. For more information, see *REST API attack types* on page 331.	Client is establishing an excessive number of TCP connections.	24	Across APIs
Content Scraping Type 1	Client abnormally accessing API content.	27	Across APIs
Content Scraping Type 2	Client abnormally accessing API content over an extended time period	28	Single API

Unauthorized client attack	Client without a token or cookie is probing an API service.	29	Single API
Single Client Login Attack Type 2	Login service attacked by a bot or rogue client over an extended time period	30	Across APIs
Header Manipulation	Probing an API using malicious headers	37	Single API

WebSocket API attack detection

ⓘ **Note:** WebSocket API attack detection is only supported when ASE is running in Inline mode.

Client identifier determination – IP address or cookie

In each 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 IP address. An environment with multiple APIs can support a mixture of identifier types in a single 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
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 API in ASE is defined by configuring `/` for `url` variable and `*` for `hostname` variable. Following is a snippet of a truncated API 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 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*

```
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

ASE and 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 blacklist reporting* on page 342.
- Delete specific client identifiers from ABS blacklist or bulk delete a type of client identifier using ABS REST API. For more information, see *Delete individual client identifiers* on page 342 and *Bulk delete client identifiers* on page 344.
- 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 *Enable or disable attack IDs* on page 345.
- 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* on page 347.

In ASE, you can:

- Manually add or delete entries from whitelist and blacklist
- 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* on page 350.

ABS blacklist reporting

ABS Provides `attacklist` REST API to complete the following two operations:

- List the various client identifiers (API Key, OAuth token, Username, Cookie, and 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

ABS provides an `attacklist` REST API with GET method to list of active attacks in the system, expired attacks, and consolidated (active and expired) attacks together. 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`

Report the 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.

Report the 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.

Report the 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`.

Delete individual client identifiers

Using the `attacklist` API with PUT method, you can delete the active client identifiers. The API requires only the body without any other headers. In the message body of the API, provide the client identifiers in their respective sections. The API checks if the client identifier is present in the active list or not before deleting. If you provide a client identifier which is not part of the active list, the API ignores such client identifiers.

URL: `/v4/abs/attacklist`

Method: PUT

Following is a sample message body for `attacklist` API to delete client identifiers:

```
{
        "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",
         ]
}
```

Following is the message showing the client identifiers that were deleted:

```
{
  "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",
        ]
    }
}
```

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 JSON file is well formed.

Bulk delete client identifiers

Use the bulk delete option when you believe that a large number of false positives have been identified. You can also use the bulk delete option to clear the blacklist in case of a reset. To bulk delete client identifiers, use the ABS `attacklist` REST API with DELETE method. Following is the URL for the API:

URL: `/v4/abs/attacklist`

Method: DELETE

To bulk delete all the entries of a client identifier or all client identifier, configure the `body` of the `attacklist` API request as show below:

```
{
 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,
}
```

In the sample request `body` above, the `attacklist` API deletes all entries for IP, Cookie, and API Key. If, in the next time interval, the AI engine flags the same client identifiers, the blacklist is populated again. 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 wan to exercise other options.
delete_all_ips	Set it true to delete all the IP addresses across all attack types from the blacklist.
delete_all_cookies	Set it true to delete all the cookies across all attack types from the blacklist.
delete_all_oauth_tokens	Set it true to delete all the OAuth token across all attack types from the blacklist.
delete_all_api_keys	Set it true to delete all the API Keys across all attack types from the blacklist.
delete_all_usernames	Set it true to delete all the usernames across all attack types from the blacklist.

Enable or disable attack IDs

You can enable or disable one or more than one attack type using ABS `attackstatus` REST API with the PUT method. 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:

- Attack ID 13: API DDoS Attack Type 2
- Attack ID 100: Decoy Attack. This attack ID can be disabled from ASE.
- Attack ID 101: Invalid API Activity. This attack ID can be disabled from ASE.

To enable or disable an attack ID, you should:

1. Use the `attackstatus` REST API with GET method to fetch the current status of an attack ID
2. Use the `attackstatus` REST API with PUT method to enable or disable the attack IDs.

Fetch the attack ID status: Run the `attackstatus` REST API with the GET method to fetch the current state of all the attack IDs. The output is divided into two sections, enabled and disabled, along with the time when an attack ID was enabled or disabled. Following is a snippet of response:

```
"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.

Disable or enable attack IDs: To disable or enable an attack ID, use the PUT method with the `attackstatus` REST API. To disable or enable an attack ID, provide the `attack_id` and `action`. The action can be `enable` or `disable`. Following is sample `body` of the PUT request:

```
{
    “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”
  }
]
}
```

Following is a sample response:

```
{
   "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 ABS AI Engine blacklist supports configuring the length of time that a client identifier type (username, OAuth token, API Key, cookie, and IP address) remains on the blacklist. 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 blacklist unless the TTL value is changed.

You can change the default value of TTL by using the `admin` ABS REST 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 blacklist that the AI engine identified from that time onwards. For example, you set initial TTL of 120-minutes at 6 AM for 100 client identifiers in the blacklist, then the list will exist till 8 AM. 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 AM. The new 30-minute TTL will apply to the client identifiers reported from 7 AM onwards.

Fetch the current TTL value: 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:

```
{
 "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",
            "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
    }
  ]
}
}
```

Configure the TTL: Use the PUT method with `admin` REST API to configure the TTL in minutes:

URL: `https://<ip>:<port>/v4/abs/admin`

Method: PUT

Body:

```
{
 "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.

Automated ASE attack blocking

Automatic blocking of attacks with ASE

When the AI Engine detects an attack, it adds an entry to its blacklist which consists of usernames, tokens, API Keys, cookies, and IP addresses of clients which were detected executing attacks. If blocking is enabled for the API, the blacklist is automatically sent to ASE nodes which blocks the client's future access using the identifiers on the list.

Activate log processing for ABS

To activate ABS log processing, execute the following ASE command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs
```

After log processing is enabled, ASE sends log data to ABS which processes the log data to look for attacks and generate reports.

Automatically block ABS detected attacks

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, execute the following ASE command to activate automatic client blocking.

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin enable_abs_attack
```

Disable attack blocking

To disable automatic sending of ABS attack lists to ASE, execute the following ASE command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin disable_abs_attack
```

Attack management in ASE

In ASE you manage detected attacks either through blacklist and whitelist. Client identifiers in blacklist are blocked by ASE while those in the whitelist are never blocked. You can also choose to block or allow a client identifier at API level by configuring the individual API JSON.

- **Whitelist** – List of “safe” 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.
- **Blacklist** – 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:
  - 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 blacklist also after further AI processing.
  - List of “bad” client identifiers manually added using ASE CLI

Manage ASE whitelist

Valid ASE operations for OAuth2 Tokens, Cookies, IP addresses, Username, and API Keys on a white list include:

- **Add an entry**

```
/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
```

```
/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
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist token
 token1.4
token token1.4 added to whitelist
```

```
/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
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_whitelist username
 user1
username user1 added to whitelist
```

- **View whitelist**

```
/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
```

- **Delete an entry**

```
/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
```

- **Clear the whitelist**

```
/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
```

Manage ASE blacklist

Valid ASE operations for IP addresses, Cookies, OAuth2 Tokens, Username, and API Keys on a black list include:

- **Add an entry**

```
/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
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist cookie
 JSESSIONID ad233edqsd1d23redwefew
cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token
 ad233edqsd1d23redwefew
token ad233edqsd1d23redwefew added to blacklist
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist api_key
 AccessKey b31dfa4678b24aa5a2daa06aba1857d4
api_key AccessKey b31dfa4678b24aa5a2daa06aba1857d4 added to blacklist
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist username
 user1
username user1 added to blacklist
```

- **View blacklist** - entire Black list or based on the type of real time violation.

```
/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
```

- **Blacklist based on decoy IP addresses**

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist decoy
Realtime Decoy Blacklist
1) type : ip, value : 4.4.4.4
```

- **Blacklist based on protocol violations**

```
/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
```

- **Blacklist based on method violations**

```
/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
```

- **Blacklist based on content-type violation**

```
/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
```

- **Automated blacklist (ABS detected attacks)**

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist
 abs_detected
No Blacklist
```

- **Delete an entry**

```
/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
```

- **Clearing the blacklist**

```
./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
```

When clearing the Blacklist, make sure that *real-time ASE detected* attacks and ABS detected attacks are disabled. If not disabled, the blacklist gets populated again as both ASE and ABS are continuously detecting attacks.

Per API blocking in ASE

ASE can be configured to selectively block on a per API basis by configuring an API JSON file parameter. To enable per API blocking for each API, set the `enable_blocking` parameter to `true` in the API JSON. For 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": ""
},
```

If per API blocking is disabled, 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`.

ASE CLI commands are also supported to enable blocking for the specified API

- `./cli.sh –u admin -p admin enable_blocking {api_id}`

Disable blocking for the specified API

- `./cli.sh –u admin -p admin disable_blocking {api_id}`

Attack reporting

Attack reports provide information about the suspected attacks on each API. The ABS Attack API provides reports by specifying the `type_id` (see descriptions in *Attack Types*) and receiving attack details including time frame, client identifier, and an attack code (see *Changing Attack Thresholds* for an explanation of attack codes). The format of the ABS `attack` API is:

`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 to *ABS 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 `login_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 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 API in a single, consolidated report, use the ABS Attack API. Attack ID 0 gives all the attacks on a single API or across APIs based on the REST API query parameters.

Consolidated attack report for an API:

The following attack API URL with attack ID as 0 gives 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`

You can further select a client identifier (IP, cookie, or a token) and carry out IP, cookie, or token forensics using the Forensic API.


```
{
 "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)"
 },
 {
 "access_time": "Fri Jan 12 09:18:34:087 2018",
 "attack_code": "varA(Tx, 25)"
 }
 ]
 },

 {
 "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)"
 },
 {
 "access_time": "Fri Jan 12 09:18:34:087 2018",
 "attack_code": "varA(Tx, 47)"
 }
 },
 },
}
```


Consolidated attack report across API:

Use the following ABS REST API to access all the attack types: `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`

You can further select a client identifier (IP, cookie, or a token) and carry out IP, cookie, or token forensics using the Forensic API.


```
[
    {
        "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)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 3)"
                    },
                    {
                        "access_time": "Tue Nov 27 16:10:00:000 2018",
                        "attack_code": "varA(Tn, 2)"
                    }
                ]
            },
            {
                "access_token": "CT27QTP01K6ZW2AK",
                "details": [
                    {
                        "access_time": "Tue Nov 27 10:50:00:000 2018",
```

```
                        "attack_code": "varA(Tn, 2)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:10:00:000 2018",
                        "attack_code": "varA(Tn, 4)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 5)"
                    }
                ]
            },
            {
                "access_token": "BDGC519O55KGG4HR",
                "details": [
                    {
                        "access_time": "Tue Nov 27 11:10:00:000 2018",
                        "attack_code": "varA(Tn, 2)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 4)"
                    },
                    {
                        "access_time": "Tue Nov 27 16:00:00:000 2018",
                        "attack_code": "varA(Tn, 2)"
                    }
                ]
            },
            {
                "access_token": "VDIFV3JH5P4VVXDW",
                "details": [
                    {
                        "access_time": "Tue Nov 27 11:30:00:000 2018",
                        "attack_code": "varA(Tn, 2)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 2)"
                    }
                ]
            },
            {
                "ip": "100.64.7.124",
                "details": [
                    {
                        "access_time": "Tue Nov 27 11:20:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:30:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)"
                    }
                ]
            },
            {
                "ip": "100.64.26.175",
                "details": [
                    {
                        "access_time": "Tue Nov 27 16:00:00:000 2018",
```

```
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)"
                    }
                ]
            },
            {
                "ip": "100.64.10.18",
                "details": [
                    {
                        "access_time": "Tue Nov 27 11:10:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)"
                    },
                    {
                        "access_time": "Tue Nov 27 11:40:00:000 2018",
                        "attack_code": "varA(Tn, 3), varA(Tn, 3)"
                    }
                ]
            }
        ]
    }
]
```

Real-time Detected attacks for inline ASE

API Security Enforcer supports real time attack detection and blocking for:

- API Pattern Enforcement – validate traffic to ensure it is consistent with the API definition
- API Deception – blocks hackers probing a Decoy API

Enable ASE detected attacks

Enable real-time ASE detected attacks by running 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
```

Disable ASE detected attacks

Disable real-time ASE detected attacks by running 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 blacklist.

In real-time, ASE blocks hackers which violate pattern enforcement or probe decoy APIs. Hacker information is reported to 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

```
{
 "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


```
{
 "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"
}
]
},
{
"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": []
}
```

Anomalous activity reporting

The Anomaly API provides detailed reporting on anomalous activity associated with a specified API. The types of anomalies detected include:

- Anomalies for each 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.
- Irregular URLs – 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 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 Deception

ASE supports configuration of decoy APIs, either the for in-context or out-of-context mode. 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.

Report ASE real-time decoy attack detection

ASE sends information about clients accessing decoy APIs to 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


```
{
 "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"
 },
 {
 "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 API URL: <ABS_IP>:port/v4/abs/decoy?earlier_date<>& later_date<>


```
{
 "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
```


See *ABS external REST APIs* for a full report.

Blocked connection reporting

ABS Blocked Connection REST API reports all connections that are blocked by 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>`

Following is a snippet of blocked connection summary report:


```
{
 "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

Following is a snippet of Blocked Connection details report:


```
{
 "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": []
 },
 {
 "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

ABS AI Engine provides in-depth information on the activities performed by a client including accessed URLs, methods, attacks, etc. The forensic report provides detailed information on the activity from an individual Token, IP address, Cookie, API key, or Username.

ⓘ **Note:** If 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.


```
{
 "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": [
 "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.


```
{
 "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"
 ]
 }
 }
}
```

Forensics on a cookie

The Cookie Forensics reports 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.

```
{
 "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
 }
 ]
 }
 ],
 "anomalies": []
 }
}
```


Forensics on API Key

The API Key Forensics reports 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"
            ]
        }
    }
}
```

Username Forensics

The username Forensics reports 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 API Metrics report provides information on client request/response activity to the requested API. It includes a summary report and detailed reporting including API access by method.

ⓘ **Note:** If 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.


```
{
 "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": []
},
{
"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
 }
 ],
 "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
 }
 ]
 }
 ]
 }
 }
}
```


Username based metrics

The username metrics report provides a summary with the total number of usernames, number of requests, tokens and IP address associated with the username. All the tokens used by the username along with the number of requests for each token is detailed.


```
{
  "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

ABS provides API key metrics including the total number of API keys and requests across all API keys. The report also lists the IP address, requesting device information, methods used, URLs accessed, and API affected. API key based metrics reporting spans all APIs.

```
{
    "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": {
                        "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
                    },
                    "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.


```
{
 "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,
"/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
 }
 }
 ]
 },
 {
 "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 URLs report includes all URLs, access count, and allowed methods for a specified API. 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
```

```
 }
```

Hacker's URL

The List Invalid URLs or hacker's URL report provide information on the four types of invalid URLs: irregular URLs, system commands, buffer overflow, and SQL injection.


```
{
 "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": [
 {
 "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"
 ]
 }
 }
}
```

Backend error reporting

The Backend Error Response Codes report provides information for each error code including client IP, server IP, and requested URL. 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"
 },
```


API DoS and DDoS threshold

API DoS and DDoS threshold 11

API Flow Control reports on API Security Enforcer 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 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:


```
{
 "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",
 "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 ABS 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)

For the Postman application, Ping Identity provides configuration files which are used by Postman to access the ABS REST API JSON information reports. Make sure to install Postman 6.2.5 or higher.

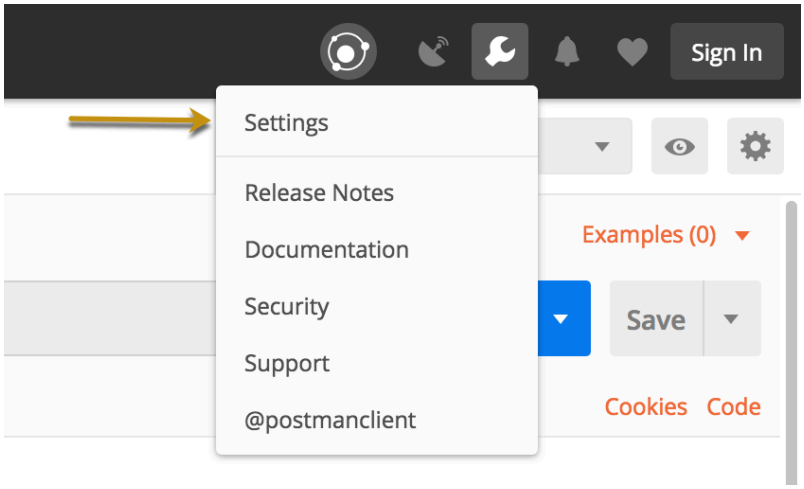
ABS self-signed certificate with Postman

ABS ships with a self-signed certificate. If you want to use Postman with the self-signed certificate of ABS, then from Postman's settings, disable the certificate verification option. Complete the following steps to disable Postman from certificate verification:

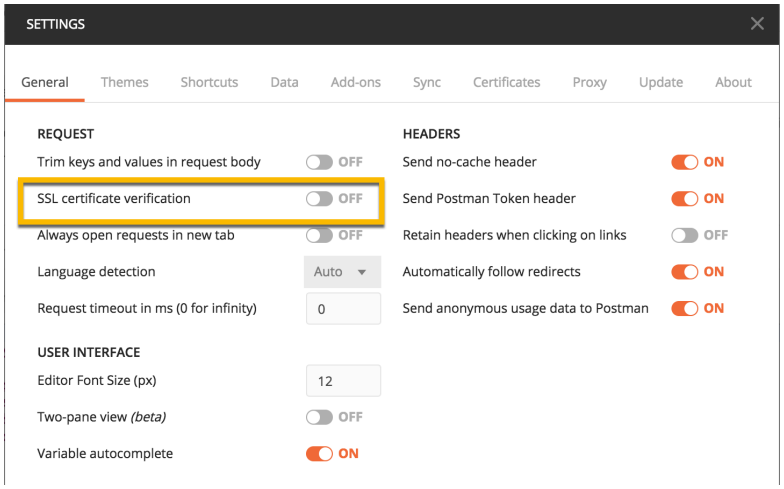
1.

Click on the **spanner**  on the top-right corner of Postman client. A drop-down window is displayed.

2. Select **Settings** from the drop-down window:



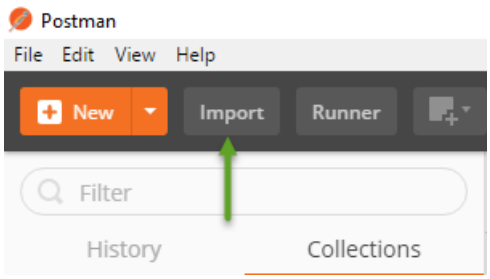
3. In the Settings window, switch-off certificate verification by clicking on the SSL certificate verification button:



View ABS reports in Postman

To view the reports, complete the following steps:

1. Download `ABS_4.1_Environment` and `ABS_4.1_Reports` JSON files from **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.



4. After importing the files, click the gear button in the upper right corner.

5. In the **MANAGE ENVIRONMENTS** pop-up window, click **ABS_4.1_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 the `opt/pingidentity/mongo/abs_init.js` for either 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 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.
   - **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.

> ⓘ **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**. *ABS external REST APIs* section provides detailed information on each API call and the JSON report response.

ABS CLI

ABS and AAD CLI provides the commands listed in the following table.

Basic commands

- Start ABS
- Stop ABS
- Help
- Update password

Obfuscation commands

- Generate obfuscation key
- Obfuscate password

Start ABS

Description

Starts ABS. Run the command from `/opt/pingidentity/abs/bin` directory

Syntax

```
./start.sh
```

Stop ABS

Description

Stops ABS. Run the command from`/opt/pingidentity/abs/bin` directory

```
./stop.sh
```

Help

Description

Displays `cli.sh` help

Syntax

```
./cli.sh help
```

Update Password

Description

Change ABS admin password

Syntax

```
./cli.sh update_password {-u admin}
```

Generate Master Key

Description

Generate the master obfuscation key `abs_master.key`

Syntax

```
./cli.sh -u admin -p admin generate_obfkey
```

Obfuscate Password

Description

Obfuscate the passwords configured in various configuration files

Syntax

```
./cli.sh -u admin -p admin obfuscate_keys
```

ABS external REST APIs

ABS external REST APIs

Following is a list of Ping Identity ABS APIs. 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:** Note that ":" (colon) is a restricted character and cannot be used in access and secret key headers in ABS external REST APIs

- *Admin API*
- *Discovery API*
- *Decoy API*
- *GET Threshold API*
- *PUT Threshold*
- *Metrics API*
- *API Key Based Metrics API*
- *OAuth2 Token Based Metrics*
- *Username Metrics*
- *Anomalies API*
- *OAuth2 Token Forensics*
- *IP Forensics API*
- *Cookie Forensics API*
- *API Key Forensics API*
- *Username Forensics API*
- *Attack Type API*
- *Flow Control API*

- *Blocked Connection API*
- *Backend Error API*
- *List Valid URLs API*
- *List Hacker's URLs API*

Admin REST API

Description: Admin API is used to fetch the list of nodes in the 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.

Method: GET

URL: `/v4/abs/admin`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response


```
{
    "company": "ping identity",
    "name": "api_admin",
    "description": "This report contains status information on all APIs, ABS clusters, a
    "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": ""
            }
        }
    ],
    "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",
                "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
            }
```

```
        ]
    }
}
```

Discovery REST API

Description: The Discovery 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

```
{
    "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"
        },
        {
            "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

Description: Decoy API provides information about the IP address that accessed the decoy 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>

Sample Response


```
{
 "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",
  "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

ABS provides Threshold REST 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* on page 317.

The following are the methods of Threshold REST APIs:

- *GET Threshold* on page 397
- *PUT Threshold* on page 398

GET Threshold

Description: The GET method in Threshold 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

```
{
    "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

```
{
    "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

Description: The PUT method in Threshold 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

```
{
 "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"
}
```

Metrics REST API

DescriptionThe Metrics 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 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.

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


```
{
 "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"
 ]
```

```
}
],
"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": [
{
"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

Description: The 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

```
{
 "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": [
{
"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

Description: The OAuth2 token-based API is used to fetch the metrics for OAuth2 token across all APIs.

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>
Secret Key	x-abs-sk	<string>

Sample Response


```
{
 "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,
"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": [
 {
 "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

Description: The Username base Metrics 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


```
{
    "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",
            "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": {
                        "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,
                    "devices": {
                        "LINUX": 35
                    },
                    "methods": {
                        "POST": 35
                    },
                    "urls": {
                        "/atm_app_oauth": 35
                    },
                    "apis": {
                        "atm_app_oauth": 35
                    }
                }
            ]
        }
    ]
}
```

Anomalies REST API

Description: The Anomalies 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

```
{
 "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": [
"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_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"
]
}
]
},
{
"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

Description: The across APIs Anomalies REST API is used to fetch the list of anomalies. 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

```
[
    {
        "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",
        "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

Description: 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


```
{
 "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": [
 "atm_app_oauth",
```

```
 "2_atm_app_oauth"
 ]
 },
 "anomaly_types": {}
 }
}
```

IP Forensics REST API

Description: The IP forensics 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

```
{
 "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"
 ]
 }
 ],
 "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

Description: Cookie forensics 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


```
{
 "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
 }
 ]
 }
 ],
 "anomalies": []
 }
}
```

Token Forensics REST API

Description: Token forensics 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

```
{
    "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": {}
    }
}
```

API Key Forensics REST API

API Key forensics 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

```
{
    "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
                    },
                    "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 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

```
{
    "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
                    }
                }
            ]
        },
        "attack_types": {},
        "anomaly_types": {}
    }
}
```


Attack Types REST and WebSocket APIs

Description: The Attack Type 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. For more information see, *REST API attacks* and *WebSocket API attacks*

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"
 ]
 }
 ]
}
```


Flow Control REST API

Description: The Flow Control 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 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


```
{
 "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

Description: The Blocked Connection 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


```
{
 "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

Description: The Backend Error 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>

	Header	**Value**
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
 "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": ""
},
{
"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

Description: The List Valid URL 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

	Header	**Value**
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
 "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
}
]
},
{
"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

Description: The List Invalid URL 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

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

```
{
 "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"
 ]
 },
 {
 "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 Blacklist REST API

Description: The Delete Blacklist REST API deletes active blacklists in ABS. The API checks if the client identifier is present in the active list or not before deleting.

Method: PUT

URL for the API: `/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”]
}
```

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"
}
```

Threshold range for Tn and Tx

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)
			REST API		
Data Exfiltration	1	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA
Single Client Login	2	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA
Multi Client Login	3	Tn = [1-100] Tx = “na”	NA	NA	NA

Stolen Cookie / Access Token	4	Tn = [2-10]	Tn = [1-19], Tx = [2-20]	NA	NA
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
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
Cookie DoS	7	Tn = [1-9] Tx = [2-10]	Tn = [1-19] Tx = [2-20]	NA	NA
API Probing Replay	8	Tn = [1-99] Tx = [2-100]	NA	NA	NA
API DoS Attack Type 1	9	Tn = [1-100] Tx = “[2-100]”	NA	NA	NA
Extreme Client Activity	10	Tn = [1-19] Tx = [2-20]	NA	NA	NA
Extreme App Activity	11	Tn = [1-19] Tx = [2-20]	NA	NA	NA
API DoS Attack	12	Tn = [1- 100] Tx = “na”	NA	NA	NA
API DDoS Attack Type 2	13	NA	NA	NA	NA
Data Deletion	14	Tn = [1- 19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA	NA
Data Poisoning	15	Tn = [1- 19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	Tn = [1-32] Tx = [2-33]	NA
Stolen Token Attack Type 2	16	Tn = [2-10] Tx = “na”	Tn = [1-100]	Tn = [1-100]	NA
Stolen Cookie Attack Type 2	17	Tn = [2-10] Tx = “na”	Tn = [1-100]	Tn = [1-100]	NA
API Probing Replay Attack 2 (client identifier: cookie)	18	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	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
API Probing Replay Attack 2 (client identifier: IP address)	20	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA
Data Exfiltration Attack Type 2	21	Tn = [1-42] Tx = [2-43]	Tn = [0-30]	Tn = [1-100]	NA
Excessive Client Connections (client identifier : cookie)	22	Tn = [1-19], Tx =[2-20]	NA	NA	NA

Excessive Client Connections (client identifier : token)	23	Tn = [1-19], Tx =[2-20]	NA	NA	NA
Excessive Client Connections (client identifier : IP address)	24	Tn = [1-19], Tx =[2-20]	NA	NA	NA
Content Scraping Type 1 (client identifier : cookie)	25	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]
Content Scraping Type 1 (client identifier : token)	26	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]
Content Scraping Type 1 (client identifier : IP address)	27	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]
Content Scraping Type 2	28	Tn = [1-29] Tx = [2-30]	Tn = [1-100]	NA	NA
Unauthorized client attack (client identifier: IP address)	29	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	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
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
Probing Replay Attack - API Key	32	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA
Extended Probing Replay Attack - API Key	33	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = 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]
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]
Sequence attack	36	Tn = [1-19] Tx = [2-20]	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
Account Takeover -UBA	38	Tn = [1-100] Tx = NA	Tn = [1-99] Tx = [2-100]	NA	NA
WebSocket API					

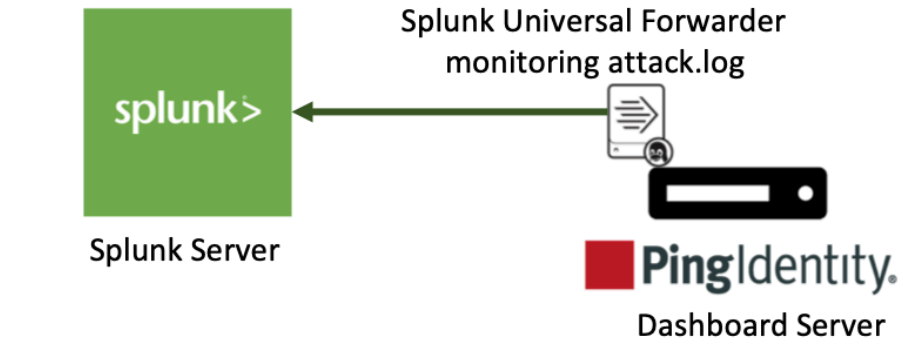
WS Cookie Attack	50	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx= [2-20]	NA	NA
WS Identity Attack	51	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA
WS DoS Attack	53	Tn = [1- 100] Tx = “na”	NA	NA	NA
WS Data Exfiltration Attack	54	Tn = [1- 100] Tx = “na”	NA	NA	NA

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 `attack.log` file in PingIntelligence for APIs Dashboard engine. Dashboard engine through `attack.log` returns a JSON report that contains attack details. Following is a snippet of the `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 Dashboard engine and Splunk server:



Splunk Universal Forwarder

ⓘ **Note:** PingIntelligence for APIs is qualified for Splunk 8.0.0.

Install and configure Splunk for PingIntelligence

About this task

Prerequisites

To complete the configuration of Splunk for PingIntelligence, you need to create a source type. Creating a source type helps Splunk to understand the event format.

Create Source type

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.

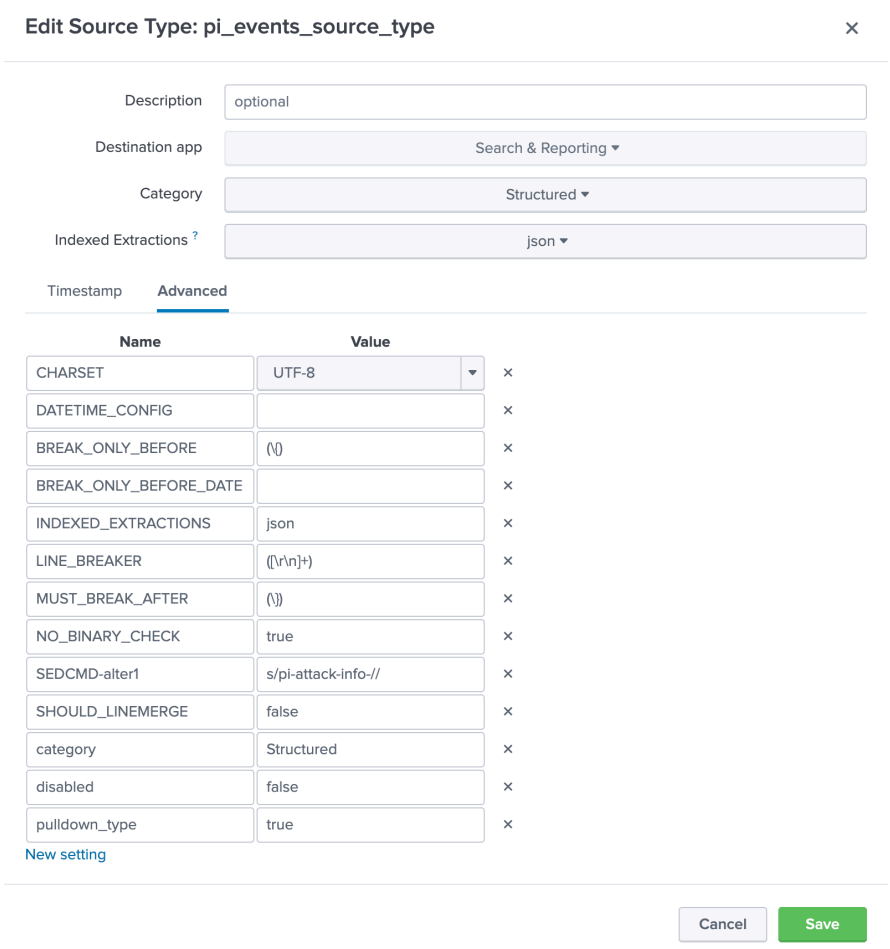
To create a source type, complete the following steps:

Steps

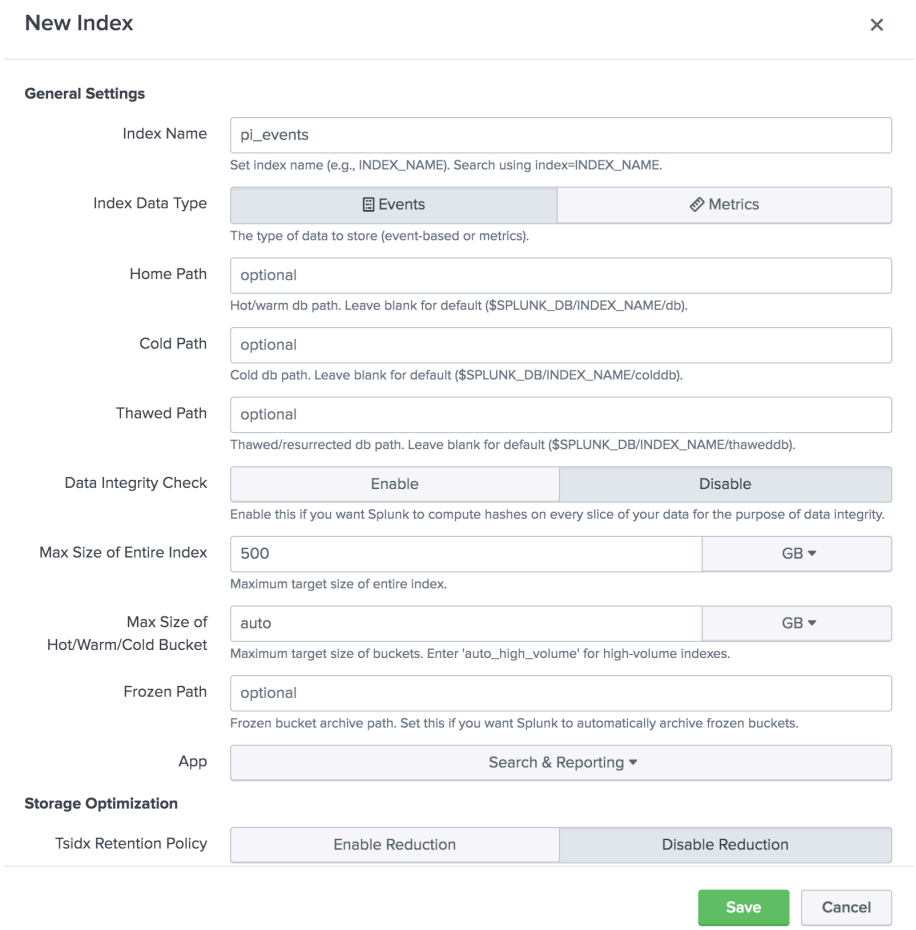
1. Configure a new source type by navigating to **Splunk Enterprise**# **Settings**# **Source Types**# **New Source type**. The source type events page is displayed.
2. Configure the **New Source type**. The fields are defined in the following table:

Name	Value
Source type name	`pi_events_source_type`
Destination app	Search and Reporting (Can change for your apps)
Category	Structures
Indexed Extractions	`json`
BREAK_ONLY_BEFORE	`(\{)`
MUST_BREAK_AFTER	`(\})`

SEDCMD-alter	`s/pi-attack-info-//`



3. Create a new index `pi_events` by navigating to **Enterprise** # **Settings** # **Indexes** # **New Index**.



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 API, MULTI_API is reported
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.

Splunk Universal Forwarder method installation and configuration

About this task

The installation and configuration process of Splunk universal forwarder method is depicted in the diagram below:

Steps

1. Install and configure Splunk Universal forwarder and start the instance using following steps:
   a. Download Splunk Universal Forwarder 8.0.0
   b. Install the Splunk universal forwarder by entering the following command. Replace the file name in the command:

```
[root@ABS]# tar -xvf splunkforwarder-8.0.0-8c86330ac18-Linux-x86_64.tgz
splunkforwarder/
splunkforwarder/share/
```

   c. Start the Splunk universal forwarder

```
[root@ABS]# cd splunkforwarder/bin
[root@ABS]# ./splunk start --accept-license
```

   d. Add forward server details (Receiver host and port in Splunk)

```
[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 above snippet in your Splunk deployment.

   e. Edit `inputs.conf` on your splunk forwarder to look like following

```
[root@ABS]# ./splunk add monitor /opt/pingidentity/splunk/data/
Added monitor of '/opt/pingidentity/splunk/data/'.
```

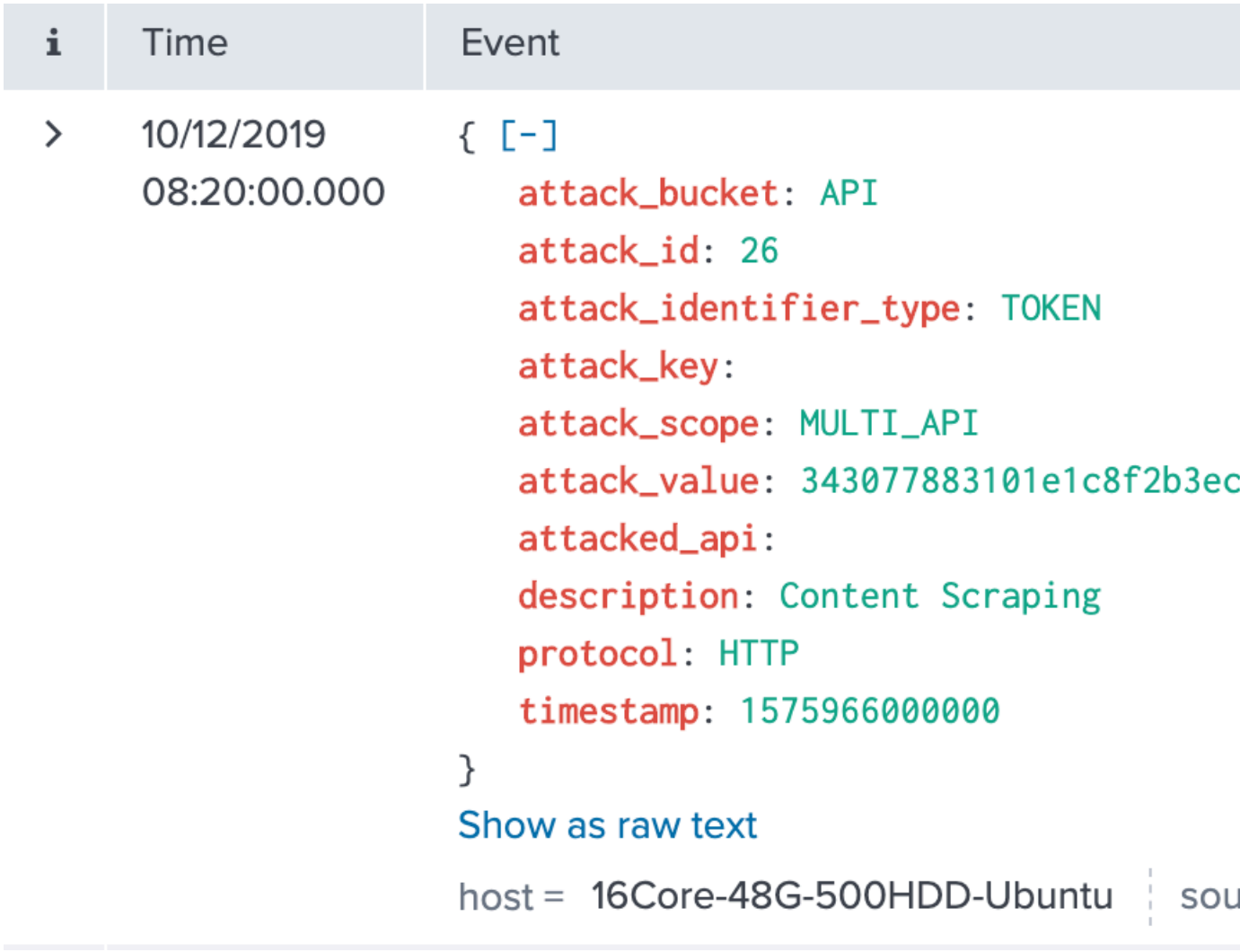
   f. Edit `inputs.conf` on your splunk forwarder

```
[root@dashboard]# cat /opt/splunkforwarder/etc/apps/search/local/
inputs.conf

[monitor:///opt/pingidentity/pingidentity/dashboard/logs/attack.log/]

index = pi_events
sourcetype=pi_events_source_type
disabled = false
```

   g. Restart Splunk universal forwarder

```
[root@ABS]# ./splunk restart
```

2. Verify if data is flowing to Splunk



> **Note:** If no data is available in Splunk, check your firewall settings.

Alert notification on Slack and Email

You can configure Splunk to send alert notification to a Slack channel or through and email.

Slack

Prerequisites:

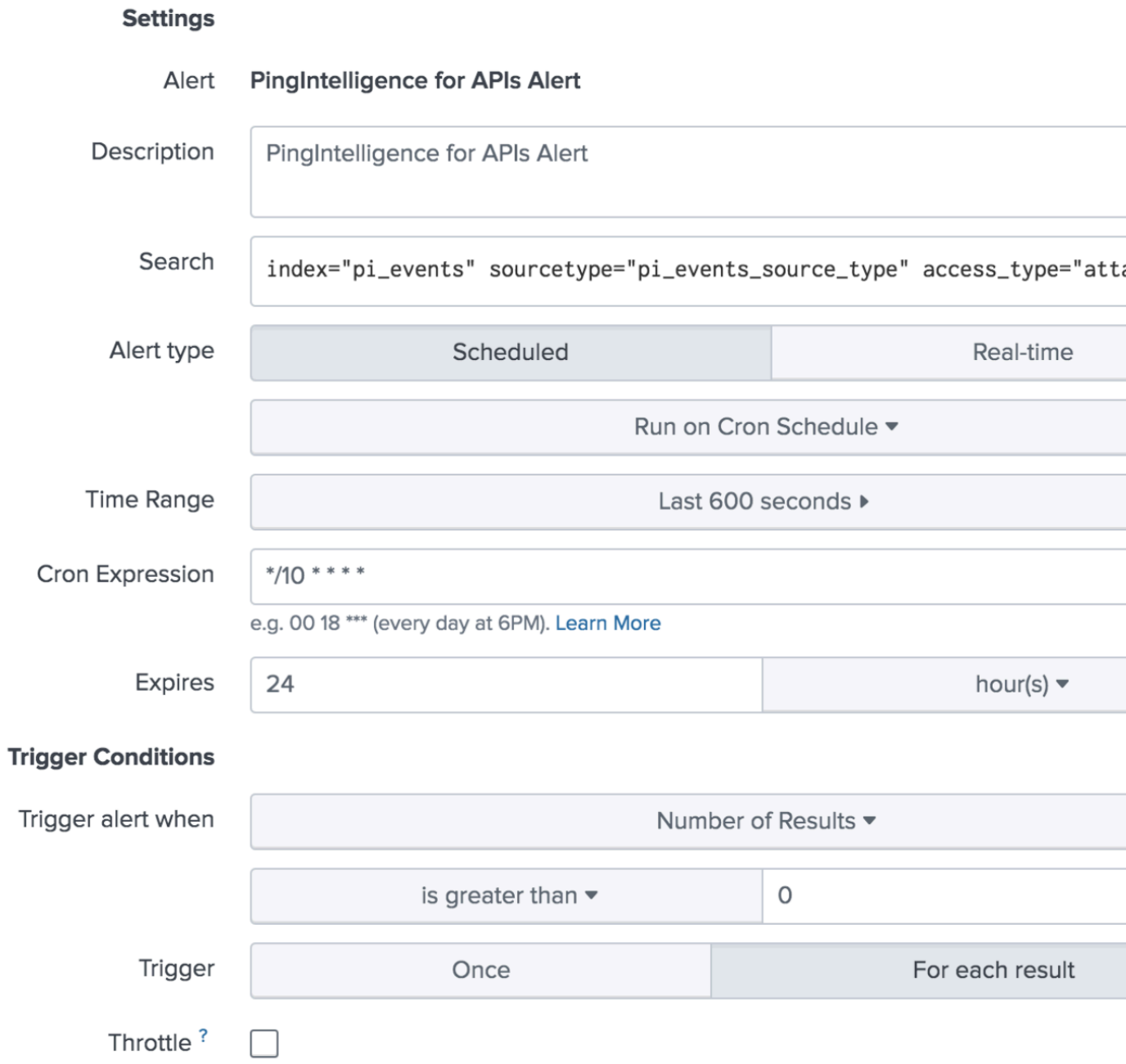
- The Slack app should already be installed in your Splunk setup.
- Connect Slack and Splunk using webhooks. For more information on Slack webhooks, see *Incoming Webhooks*

Complete the following steps to create an alert for Slack:

1. Navigate to **Settings** #> **Searches, reports and alerts**

> **Note:** Alert should be created for App: Search & Reporting(search)

2. Create new alerts

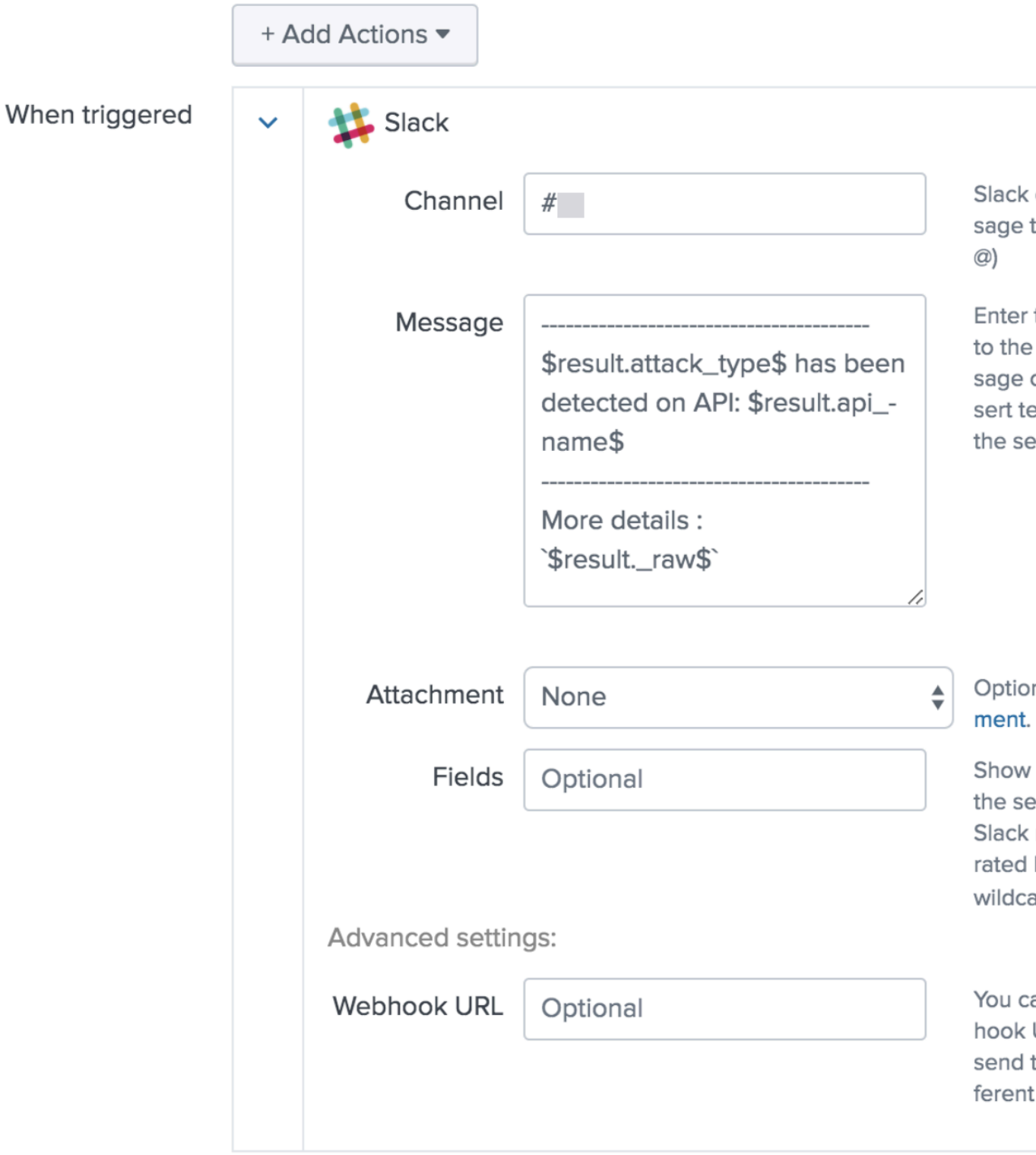


Enter the values as described in the table below:

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**
Cron Expression	*/10 * * * *

Time Range	600
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



Value	Description
Add Actions	Choose the slack app to add actions
Channel	Use the channel which has been configured with webhook URL which starts with either # or @ In this example, we are using channel name as: # PingIntelligence_alerts

Message	This is the message which will be posted along with the alert in slack, We recommend using the below message:
	`----------`
	`$result.attack_type$ has been detected on API: $result.api_name$`
	`----------`
	`More details :`
	`` `$result._raw$` ``

Attachments	NA
Fields	NA
Webhook URL	NA

4. Post a message in Splunk to verify that it is notified in Slack

ABS log messages

The following tables list the critical log messages from `abs.log` and `aad.log file`. `abs.log` file is rotated every 24-hours. For more information, see *ABS logs* on page 297

abs.log mesaages:

Log message	Description
Warn :-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 *ABS License and timezone* on page 281
Warn :- Attempt to shutdown ABS from 127.0.0.1	This message is logged in `abs.log` when shutdown of ABS AI engine is initiated.
Warn :- 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 blacklist.
Warn :- 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 blacklist
Warn :- 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 blacklist.

Log message	Description
Warn :- 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 blacklist.
Warn :- License is Expired. Please renew your license	This message is logged in `abs.log` when ABS license has expired. For more inforamtion, see *ABS License and timezone* on page 281
Warn :- MongoDB primary node is down	This message is logged in `abs.log` when a MongoDB connection failure occurs.
Warn :- Stream init-wait interrupted	This message is logged in `abs.log` when streaming of access log files is interrupted.
Warn :- 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.
Warn :- Error while closing mongo connections	This message is logged in `abs.log` when shutdown of MongoDB connection was not successful.
Warn :- Error while loading anomaly dictionary from mongo	This message is logged in `abs.log` when writing of anomalies to data directory fails.
Warn :- 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.

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, deobfuscation 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.
error while loading restricted user auth keys from metadata db in mongo	This message is logged in `abs.log` when MongoDB is not accessible.

Log message	Description
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.

PingIntelligence Dashboard

Introduction

PingIntelligence for APIs Dashboard offers the following:

- View various APIs managed by PingIntelligence
- View the training status of your APIs.
- Hide or display APIs from the main API dashboard, sort APIs or search for APIs
- View the API dashboard
- Unblock and update thresholds for a blocked client

The dashboard engine utilizes Elasticsearch and Kibana to provide a graphical view of an API environment including user activity, attack information, and blocked connections. The dashboard engine fetches data from ABS AI engine. This data from the ABS AI engine is rendered in Web GUI.

Installation prerequisite

Ensure that the following prerequisites are completed before installing PingIntelligence Dashboard:

- **Server**: 8 core CPU, 16 GB, 1 TB HDD
- **Operating system**: RHEL 7.6 or Ubuntu 16.0.4 LTS
- **OpenJDK**: 11.0.2 to 11.0.6
- **SSL certificate**: One private key and certificate. By default, PingIntelligence Dashboard uses the private key and certificate shipped with the binary.
- **Password**: To change the default password, set a minimum 8 character password
- **ABS**: ABS AI engine URL, access, and secret key. Make sure that ABS is reachable from PingIntelligence Dashboard.
- **ASE**: ASE management URL, access, and secret key. Make sure that ASE is reachable from the PingIntelligence Dashboard.

ⓘ **Note:** Connecting Dashboard to ASE is optional. Functionality like adding discovered APIs to ASE and attack management will be limited.

Port numbers: The following is a list of default port numbers. Make sure that these are available for installing PingIntelligence Dashboard.

- **PingIntelligence Dashboard server**: 8030
- **Elasticsearch**: 9200
- **Kibana**: 5601
- **H2 database**: 9092. H2 database is installed and runs as a part of PingIntelligence Dashboard.

Operating system configurations: Modify the following settings 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
```

- **JDK installation**: Set `JAVA_HOME` to `<jdk_install>` directory and add `<jdk_install>/bin` to system `PATH` variable
- Choose the `<pi_install_dir>` directory. `<pi_install_dir>` should be readable and writable by the logged in user.

PingIntelligence Dashboard users: The two pre-configured login users in PingIntelligence Dashboard. The two users are:

- `admin`
- `ping_user`

Multiple users can share the `admin` and `ping_user` logins simultaneously on PingIntelligence Dashboard. The admin user has access all to PingIntelligence Dashboard functions. An `admin` user can view all Dashboards, manage blocked clients, access API definitions, and view license information. A `ping_user` can only view all the API dashboards. A total of 25 `admin` and `ping_user` can log in simultaneously.

Browser support

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 49.0 and later	Version 69.0 and later	Version 12.0 and later	
Mac OS Sierra -10.12	Version 49.0 and later	Version 69.0 and later	Version 10.1 and later	
Mac OS High Sierra - 10.13	Version 49.0 and later	Version 69.0 and later	Version 11.1 and later	
Mac OS Catalina -10.15	Version 49.0 and later	Version 69.0 and later	Version 13.0 and later	
Windows 7	Version 49.0 and later	Version 69.0 and later		
Windows 8.0	Version 49.0 and later	Version 69.0 and later		

Operating System	Google Chrome	Mozilla Firefox	Apple Safari	Microsoft Edge
Windows 8.1	Version 49.0 and later	Version 69.0 and later		
Windows 10	Version 49.0 and later	Version 69.0 and later		Version 42.7, 44.7, 79.0, 80.0 and later

Install PingIntelligence Dashboard

Complete the following steps to install PingIntelligence Dashboard:

1. Create a `ping_install_dir` directory on your host machine
2. *Download* the PingIntelligence Dashboard binary
3. *Download* Elasticsearch 6.8.1
4. *Download* Kibana 6.8.1
5. Change directory to `ping_install_dir`:

```
# cd pi_install_dir
```

6. Untar the PingIntelligence Dashboard:

```
# tar -zxf pi-api-dashboard-4.1.1.tar.gz
```

7. Change directory to `/pingidentity/webgui/`

```
# cd pingidentity/webgui/
```

8. Install PingIntelligence Dashboard by entering the following command and follow the instructions displayed on the prompt:

```
# ./bin/pi-install-ui.sh
```

```
elasticsearch-6.8.1.tar.gz file path >
kibana-6.8.1-linux-x86_64.tar.gz file path >

Use bundled ssl key and self signed certificate for ui server [y/n]? >[n]
ssl private key path >
ssl certificate path >

Use default password [changeme] for all components and users [y/n]? > [n]
UI login admin user 'admin' password >
Renter UI login admin user 'admin' password >
UI login regular user 'ping_user' password >
Renter UI login regular user 'ping_user' password >

ABS url >
Use default access/secret key for ABS  [y/n] ? > [n]
ABS access key >
ABS secret key >

ASE management url >
Use default access/secret key for ASE  [y/n] ? > [n]
ASE access key >
ASE secret key >


configuring elasticsearch… please wait for 15 seconds
```

```
elasticsearch config is completed.

configuring kibana…please wait 60 seconds
kibana config is completed.

configuring dashboard…
generating new obfuscation master key
dashboard config is completed.

configuring webgui…
generating new obfuscation master key
webgui config is completed.

saving auto generated credentials for all components to
 webgui_internal.creds file

WebGUI installation completed.

Start WebGUI [y/n] > [y]

start elasticsearch...
 elasticsearch started. Log is available at elasticsearch/logs/
elasticsearch.log

start dashboard……
 dashboard started. Log available at dashboard/logs/dashboard.log

start kibana……
  kibana started. Log available at kibana/logs/kibana.log

start ui server……
  UI server started. Log available at webgui/logs/admin/admin.log

WebGUI started. Log available at webgui/logs/admin/admin.log

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.

Important Action:
1) Credentials for all internal components are available in
 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) Two obfuscation master keys are auto-generated
   pingidentity/webgui/config/webgui_master.key
   pingidentity/dashboard/config/dashboard_master.key
3) For security purposes you should move obfuscation master keys from this
 server. But when components
   are restarted, master keys should be present at the original locations.
```

Verify the installation

You can verify the installation by checking the process IDs (PID) of each component. You can check the `pid` of components at the following location:

- **Elasticsearch**: `<pi_install_dir>/elasticsearch/logs/elasticsearch.pid`
- **Kibana**: `<pi_install_dir>/kibana/logs/kibana.pid`
- **Dashboard**: `<pi_install_dir>/dashboard/logs/dashboard.pid`

Tune Dashboard performance parameters

Configure the following three parameters for Dashboard's better performance.

Parameter	Description	Location
Elasticsearch		
-Xms and -Xmx	▪ **Xms** - Defines the minimum heap size of Elasticsearch. Set it to 4GB as `Xms4g`. ▪ **Xmx** - Defines the maximum heap size of Elasticsearch. Set it to 4GB as `Xmx4g`.	`$ES_HOME/config/jvm.options`
thread_pool.search.size	Defines thread pool size for count/search/suggest operations in Elasticsearch. Configure it to 50% of total CPUs allocated.	`$ES_HOME/config/elasticsearch.yml`
Kibana		
elasticsearch.requestTimeout	Time (in milliseconds) to wait for Elasticsearch to complete the request and return the response back to Kibana. Set the value to 60000 milliseconds.	`$kibana_HOME/config/kibana.yml`

ⓘ **Note:** To detect and mitigate attacks like Cross Site Scripting(XSS), 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;
```

Start and stop Dashboard

You can choose to start and stop all the components together or individually. 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.. [started]

Verifying elasticsearch connectivity.. [OK]
Verifying ABS connectivity.. [OK]

Starting dashboard.. [started]
Starting kibana.. [started]


Verifying Kibana connectivity.. [OK]
Verifying ASE connectivity.. [OK]

Starting webgui.. [started]
```

```
WebGUI started.
```

To stop all the components of PingIntelligence Dashboard together, enter the following command:

```
# cd <pi_install_dir>/pingidentity/webgui
# ./bin/stop-all.sh

Stopping webgui..[stopped]
Stopping dashboard.. [stopped]
Stopping  kibana.. [stopped]
Stopping  elasticsearch.. [stopped]

WebGUI stopped.
```

Start and stop PingIntelligence Dashboard components individually

Start the components in the following order:

1. **Start Elasticsearch**: Enter the following command to start Elasticsearch:

   ```
   # cd <pi_install_dir>/pingidentity/elasticsearch
   # ./bin/elasticsearch -d -p logs/elasticsearch.pid
   ```

2. **Start Dashboard**: Enter the following command to start Dashboard:

   ```
   # cd <pi_install_dir>/pingidentity/dashboard
   # ./bin/start.sh
   ```

3. **Start Kibana**: Enter the following command to start Kibana:

   ```
   # cd <pi_install_dir>/pingidentity/kibana
   # ./bin/kibana >> ./logs/kibana.log 2>&1 & echo $! > logs/kibana.pid
   ```

4. **Start Web GUI**: Enter the following command to start Web GUI:

   ```
   # cd <pi_install_dir>/pingidentity/webgui
   # ./bin/start.sh
   ```

Stop the components individually by entering the following commands:

Stop Elasticsearch: Stop Elasticsearch by entering the following command:

```
# cd <pi_install_dir>/pingidentity/elasticsearch
# kill -15 "$(<logs/elasticsearch.pid)"
```

Stop dashboard engine: Stop the dashboard engine by entering the following command:

```
# cd <pi_install_dir>/pingidentity/dashboard
# ./bin/stop.sh
```

Stop Kibana: Stop Kibana by entering the following command:

```
# cd <pi_install_dir>/pingidentity/kibana
# kill -9 "$(<logs/kibana.pid)"
```

Stop Web GUI: Enter the following command to stop Web GUI:

```
# cd <pi_install_dir>/pingidentity/webgui
# ./bin/stop.sh
```

Start and stop PingIntelligence Dashboard as a service

You can also start and stop PingIntelligence Dashboard as a service. Complete the following steps to start PingIntelligence Dashboard as a service:

1. Navigate to the `util` directory and run the following command to install PingIntelligence Dashboard as a service:

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

2. Start the service by entering the following command:

```
systemctl start pi-webgui.service
```

To stop PingIntelligence Dashboard service, run the following command:

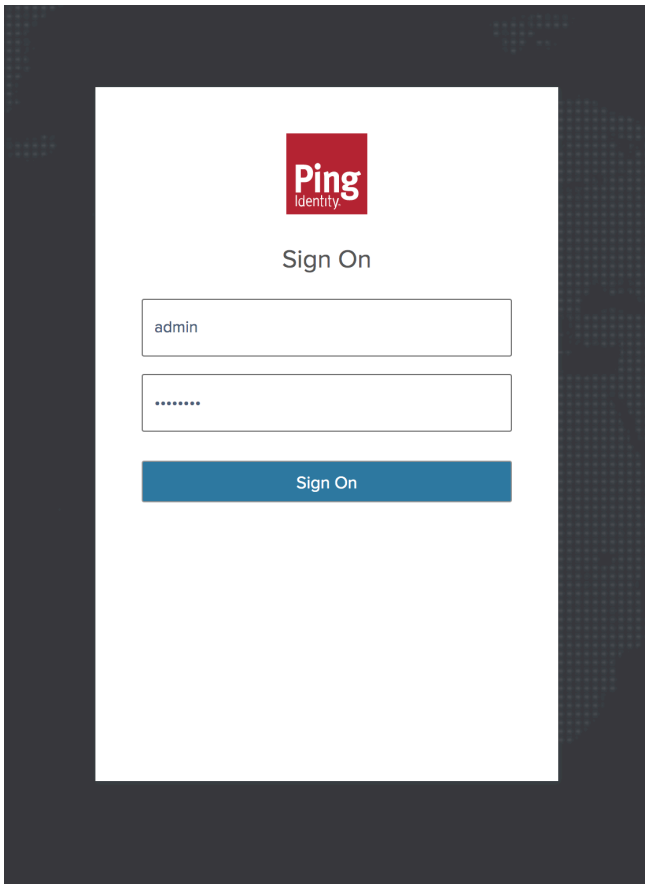
```
systemctl stop pi-webgui.service
```

Access PingIntelligence Dashboard

Access the PingIntelligence for APIs Dashboard from a browser at this default URL: `https://<pi_install_host>:8030`.

At the login screen, login 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 PoC deployment use the default password.



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

PingIntelligence Dashboard is categorized into the following components:

- **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**

Configure Kibana to prevent clickjack attacks

Clickjacking is a method used by hackers to trick you on clicking on something different than what you may perceive. To prevent Clickjacking attack in PingIntelligence Dashboard, configure Kibana with an **X-Frame-Options** header so that it cannot be loaded from third-party pages or an iframe.

Complete the following steps to add the header to the Kibana configuration:

1. Navigate to the `<pi_install_dir>/kibana/config` directory.

   ⓘ **Note:** Substitute <pi_install_dir> with the default PingIdentity directorypath.

2. Edit the `kibana.yml` file to, add **server.customResponseHeaders: {"x-frame-options":"sameorigin"}** at the end of the file. Save the file.

```
# vi config/kibana.yml
```

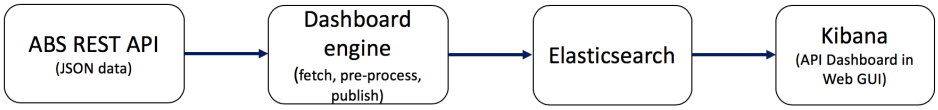
The following is a sample `kibana.yml` file.

```
server.ssl.enabled: true
server.ssl.key: config/ssl/kibana.key
server.ssl.certificate: config/ssl/kibana.crt
server.ssl.supportedProtocols: ["TLSv1.2"]
elasticsearch.ssl.verificationMode: none
elasticsearch.username: "xxxxxxxxx"
elasticsearch.url: "https://127.0.0.1:1234"
elasticsearch.password: "xxxxxxxxx"
server.host: "0.0.0.0"
server.port: "5678"
server.defaultRoute: "/app/kibana#/dashboard/pingapiintelligence"
server.basePath: "/pi/ui/dashboard"
server.rewriteBasePath: true
server.customResponseHeaders: {"x-frame-options":"sameorigin"}
```

Dashboard

The Dashboard provides a near real-time snapshot of your API environment. It provides insights on user activity, attack information, blocked connections, forensic data, and much more. The Dashboard makes periodic REST API calls to the ABS (API Behavioral Security) AI engine, which returns JSON reports that

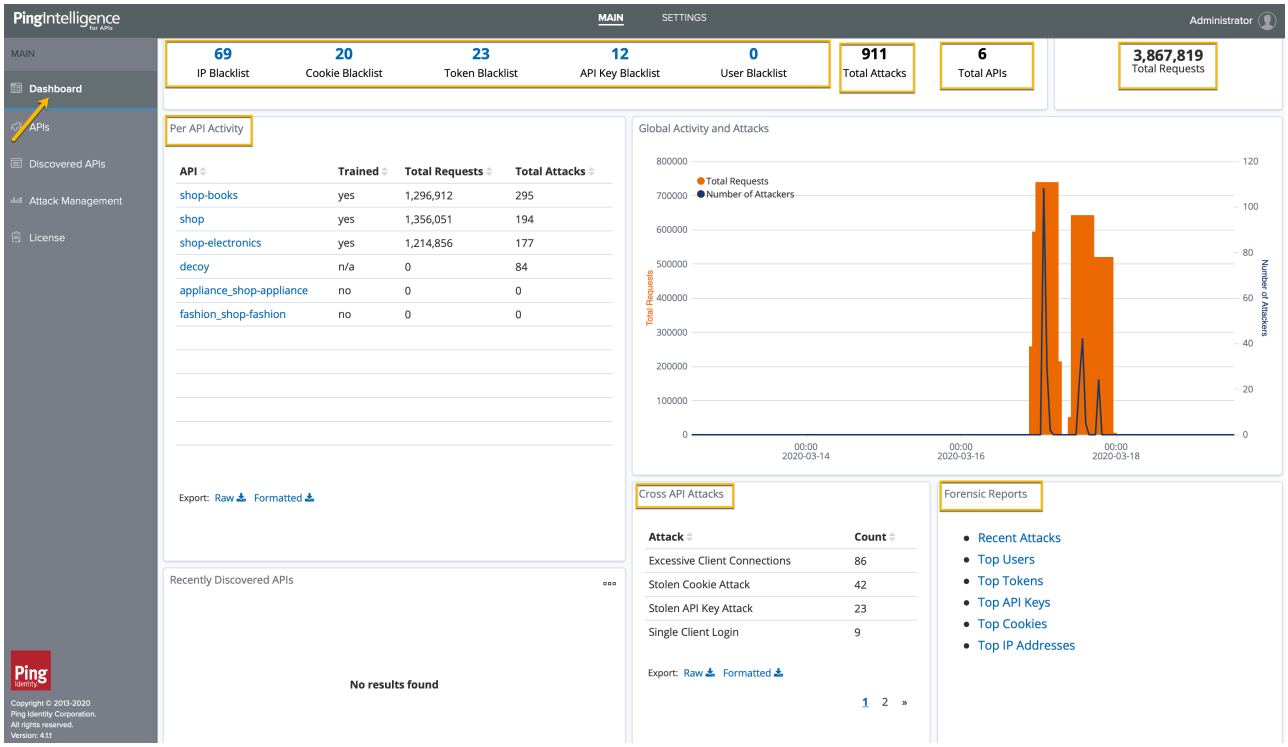
are used to generate visualizations and API metrics. The following illustration shows the data flow for API



dashboard.

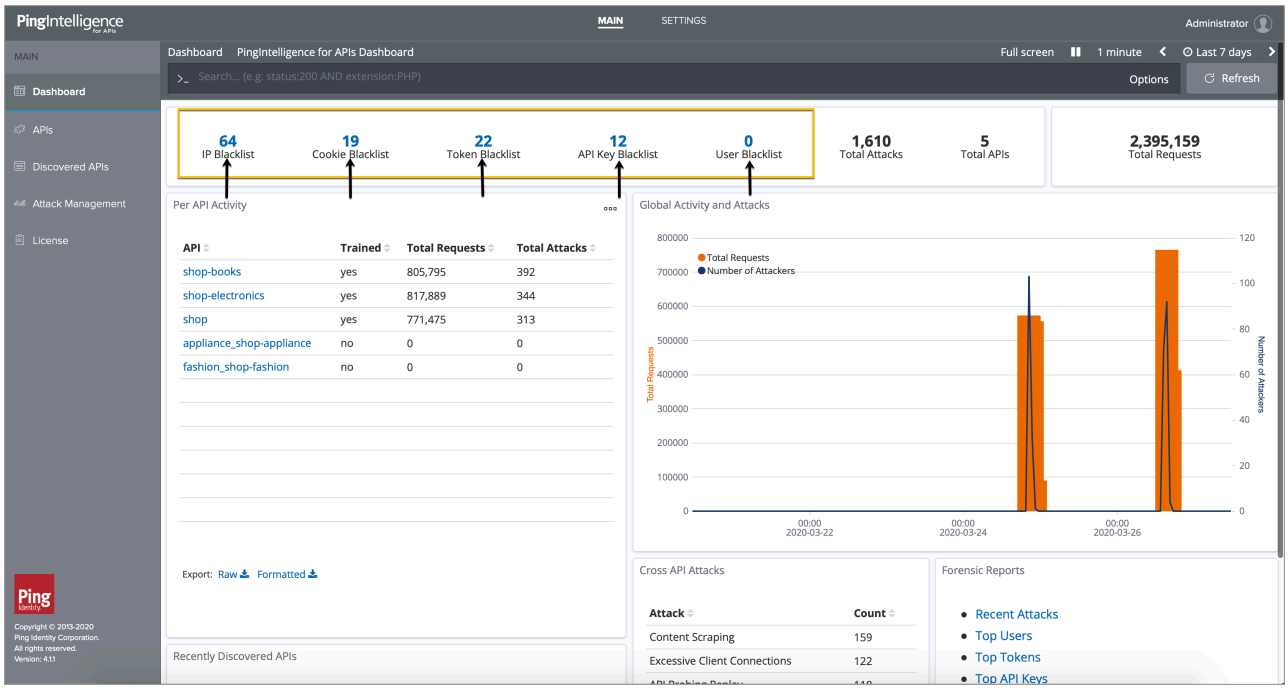
To view the API dashboard, click on **Dashboard**. The Dashboard provides information on the following::

- Global metrics like:
  - Blacklist across APIs for each client identifier. For more information, see *Interactive blacklists* on page 22.
  - Total attacks across APIs
  - Total requests across APIs
  - Number of APIs in your environment
- Time series visualization of total number of requests and attacks. For more information, see *Dashboard time series* on page 24.
- Data on Per API activity. For more information, see *Per API activity* on page 457.
- Data on attacks across APIs. For more information, see *Cross API attacks and recently discovered APIs* on page 464.
- Forensic reports across APIs. For more information, see *Forensic reports* on page 459.
- Recently discovered APIs in the environment.



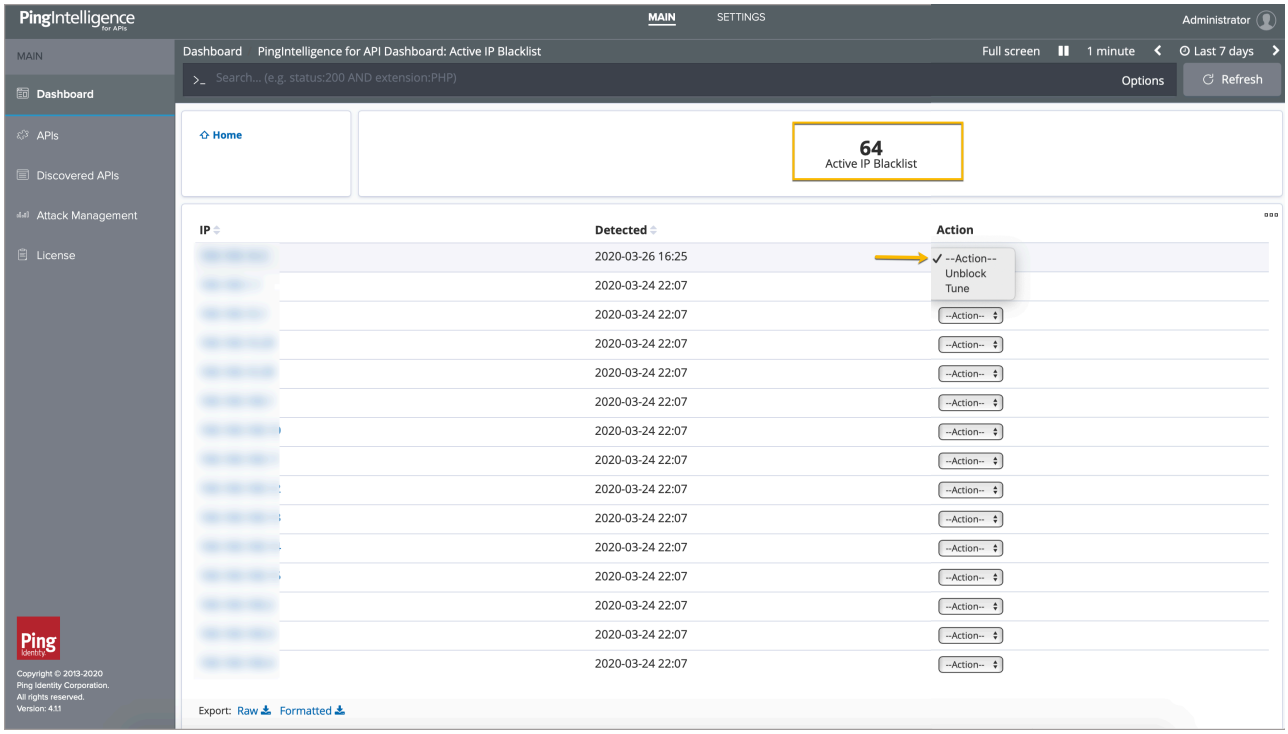
Interactive blacklists

PingIntelligence for APIs Dashboard provides the capability of interactive blacklist management. A blacklist is a list of client identifiers that were detected executing an attack. The dashboard enables you to unblock the blacklisted client identifiers or tune the threshold values for attack types. It supports the following client identifier types- IP address, Cookie, Token, API Key, and Username. You can view the top-500 entries on each blacklist from the dashboard.



Click on the count for any blacklist type, for example, **IP Blacklist**. The dashboard lists the blacklisted IP addresses along with the Detected date..

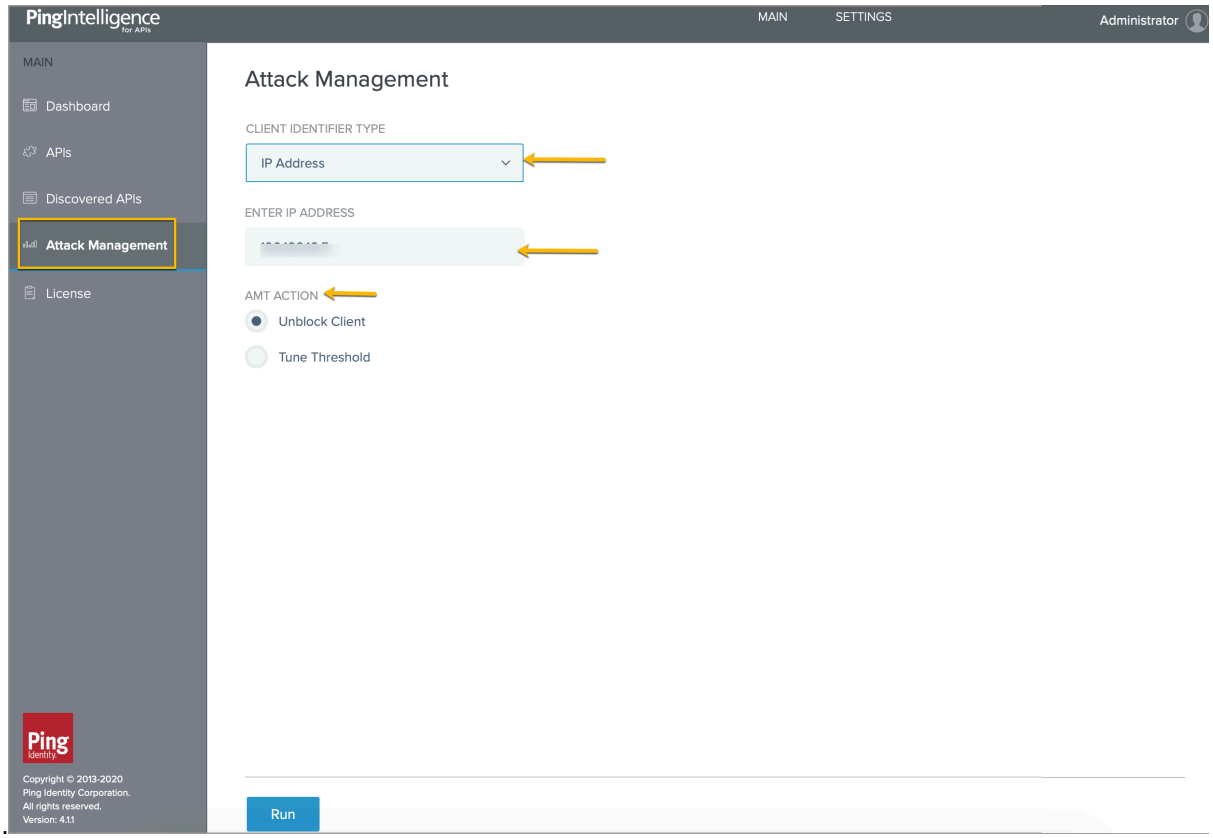
The following screenshot shows the expanded blacklist:



For each blacklisted IP address, you get the option to Unblock or Tune in the **Action** list. Clicking on either action redirects the dashboard to the Attack management application. Attack management allows you to run the operations for unblocking the client identifiers and tuning the threshold values.

ⓘ **Note:** The **Action** list is available only for an Admin user. You need to have Admin user privileges to perform**Unblock** and **Tune** operations on a client identifier.

The following screen shot shows the Attack management UI.



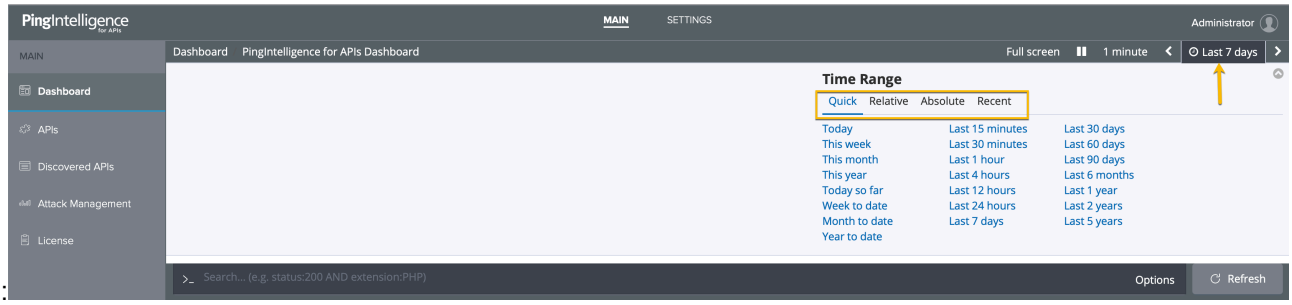
The values in **Client Identifier Type** and **Enter IP Address** get auto-populated into the Attack management application from the dashboard. The **AMT Action** is auto-selected. Click **Run** to execute the operation. For more information on Attack management, see *Attack management* on page 19.

> ⓘ **Note:** Dashboard does not populate the API key key-name in the Attack management application when the client identifier is API key. It only populates the API key value.

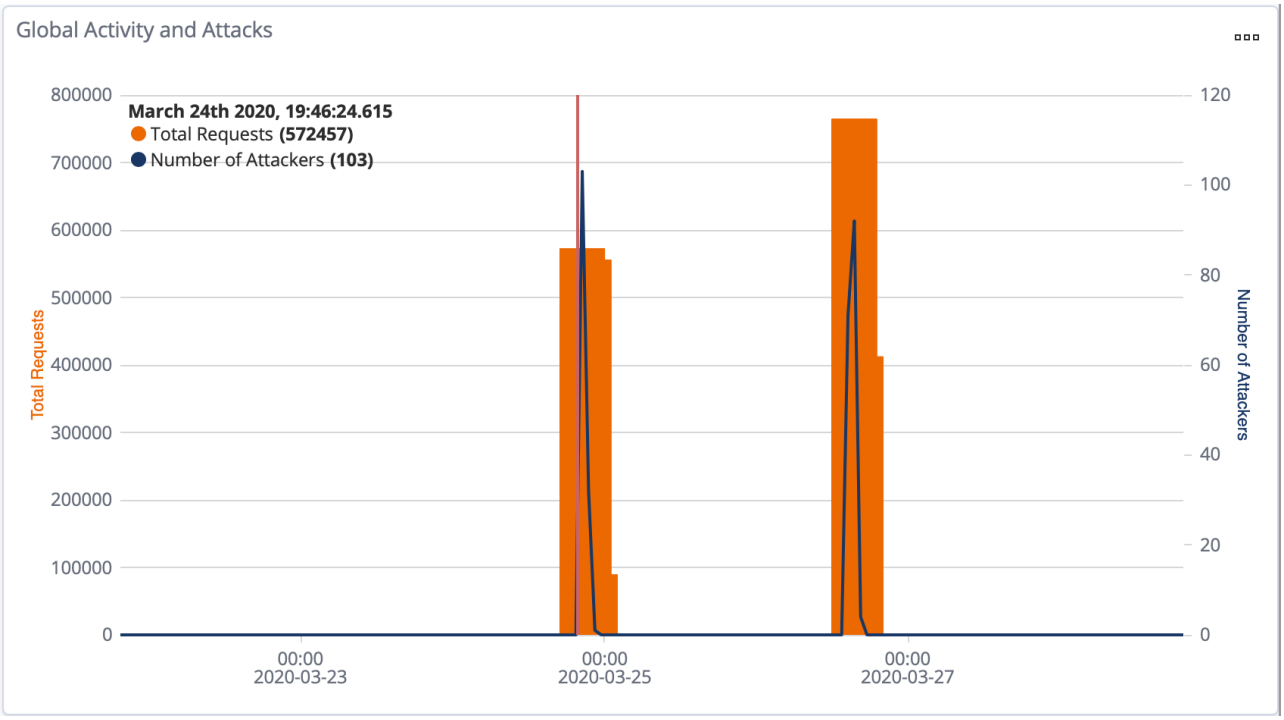
Dashboard time series

PingIntelligence Dashboard shows the attacks in a time-series format. To adjust the timeframe viewed on the Dashboard, click between the **time-period** arrows located on the top right corner of the dashboard and select the desired time period.

See the example in the following screen capture:



The following screen capture shows the total requests and number of attackers data in time series format.

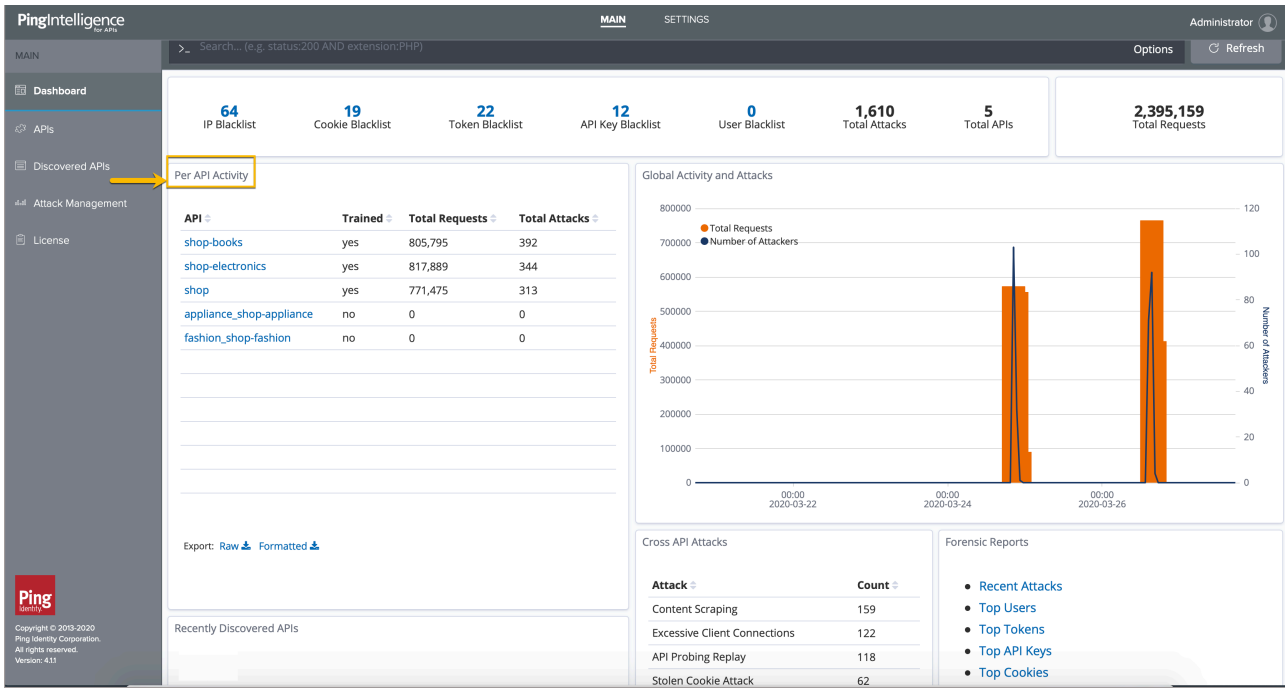


Per API activity

PingIntelligence for APIs Dashboard provides detailed analytics on each API. On the main Dashboard, the following information is available:

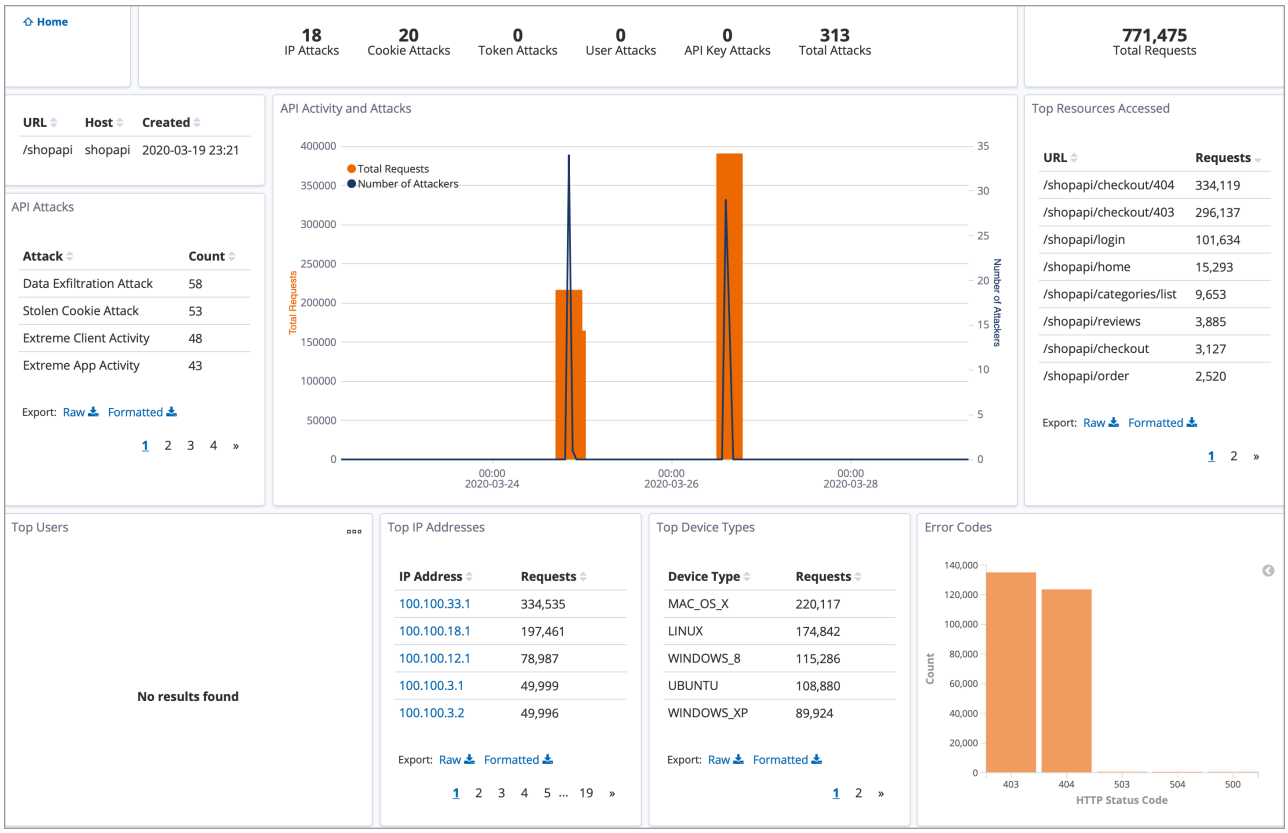
- Training status - An API is trained after the ABS AI engine analyzes its traffic patterns and builds AI models to detect attacks on the API. For more information, see *AI Engine training* on page 312
- Total number of requests made to the API during the requested timeframe
- Total attacks on the API during the requested timeframe

.



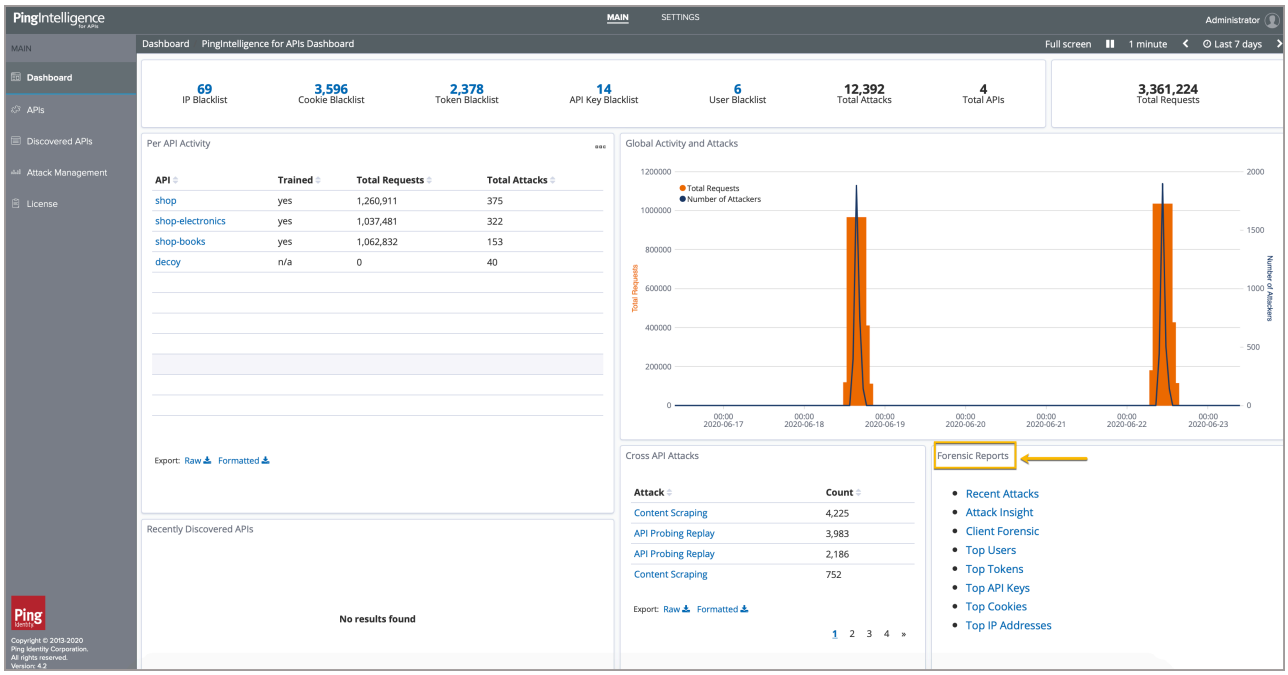
Click on the API name to launch an API-specific dashboard. The dashboard provides following insight into the API activity:

- Client attacks on the API - PingIntelligence for APIs identifies the number of individual clients executing attacks by client identifier - API Keys, Cookies, IP addresses, OAuth2 tokens, and Usernames. It also displays the total number of attacks (including multiple attacks per client) originating from any client identifier
- API Attacks - ABS AI Engine reports on client attacks targeted on a specific API. It identifies different attack types on your API based on client activity. The dashboard displays information that can be sorted by attack type or count. For more information, see *REST API attacks* on page 330.
- API Activity and Attacks - The API dashboard provides the total number of requests and attackers for the API in a time-series format.
- Top Resources Accessed - The most frequently accessed API resources can be viewed in sorted order by URL or number of requests.
- Top IP Addresses - The IP addresses from which the API requests have originated can be viewed in sorted order by IP or number of requests.
- Top Device Types – The device type from which the API requests have originated can be viewed in sorted order by device type or the number of requests.
- Top Users – The username accessing the API can be viewed in sorted order by username or number of requests.
- Error Codes - The number of failed requests are categorized by HTTP status codes in a bar chart.

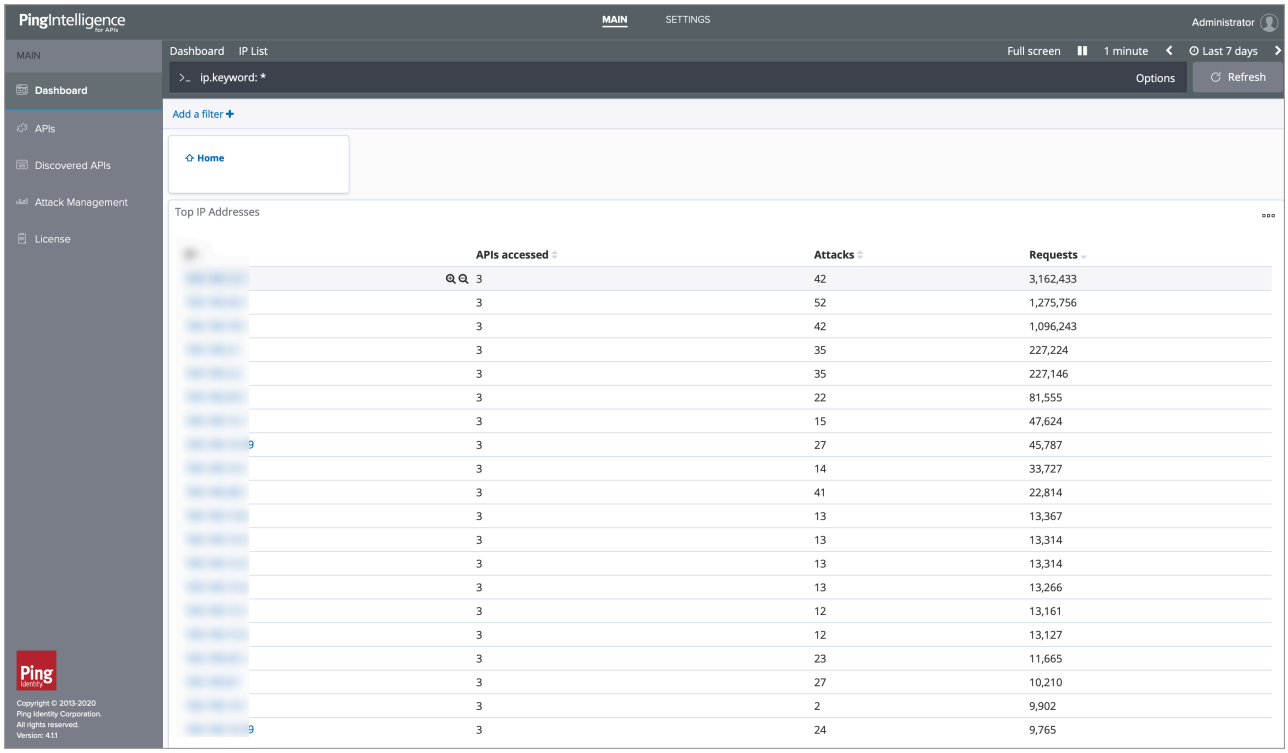


Forensic reports

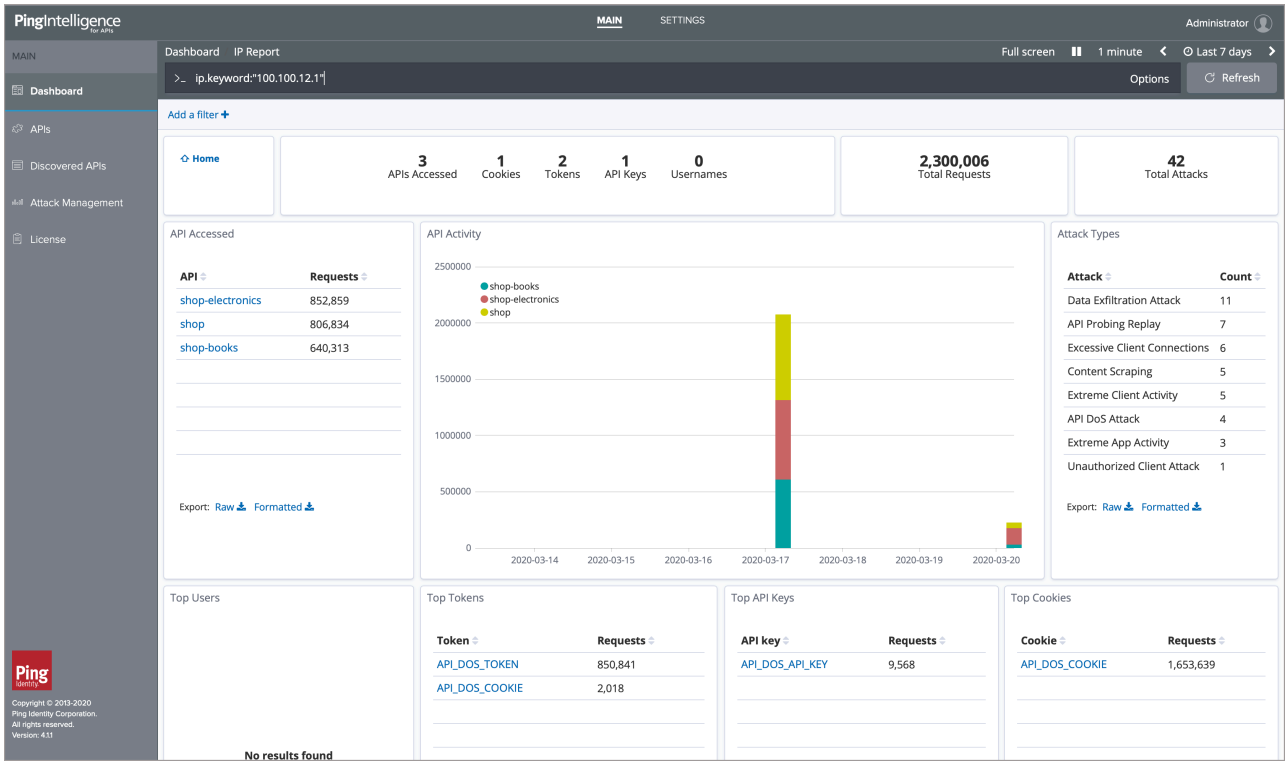
The Forensic reports provided by PingIntelligence for APIs Dashboard render deep insights into the client API activity. They provide analytics on recent attacks and API activities of top clients. The ABS AI Engine generates forensic insights by analyzing the API traffic patterns. PingIntelligence for APIs Dashboard projects these insights as Forensic reports.



An example of top clients is the Top IP Addresses forensic report, which gives insight into the activity of the most common IP addresses accessing APIs in your environment.



Click on the individual IP addresses to drill down further and get details on the APIs accessed by the IP address, attacks generated by it and so on.



You can get similar insights on activites from other client identifiers including Usernames, API Keys, Tokens, and Cookies.

ⓘ **Note:** For client identifiers that exceed 4096 characters, the Dashboard displays only attack information. It does not report forensic details, such as APIs accessed by the identifiers, number of requests, and so forth.

The following table shows the forensics reported by the Dashboard.

Forensic reports	Analytics reported
Recent Attacks	The forensic report on recent attacks give following information: ▪ The time of the attack. ▪ The API attacked. ▪ The client identifier from which the attack originated - The identifier can be IP address, cookie, token, username, API key. ▪ Count of attacks.
▪ Top Users ▪ Top Tokens ▪ Top API Keys ▪ Top Cookies ▪ Top IP Addressess	The Top client identifier reports provide a list of following information for each client identifier: ▪ Client identifier value. ▪ Count of APIs attacked using the client identifier. ▪ Number of attacks from the client identifier. ▪ Number of requests made using the client identifier.

ⓘ **Note:** You can drill down on each client identifier to get more finer insights like APIs accessed, count of requests made to each API, type of attacks generated from the client identifier, devices from which the request initiated and much more.

With PingIntelligence Dashboard, you can get the attack insights for different time-periods. For more information, see *Dashboard time series* on page 24.

Client forensic report

The Client Forensic report provides insights into client activity in the course of an attack. It presents a detailed analysis of the client traffic patterns prior to an attack. The report gives the following information :

- The APIs accessed by the client and the total number of requests made to these APIs.
- The different types of attacks executed by the client and the count of those attacks.
- The URLs accessed by the client and the total number of requests made to these URLs.

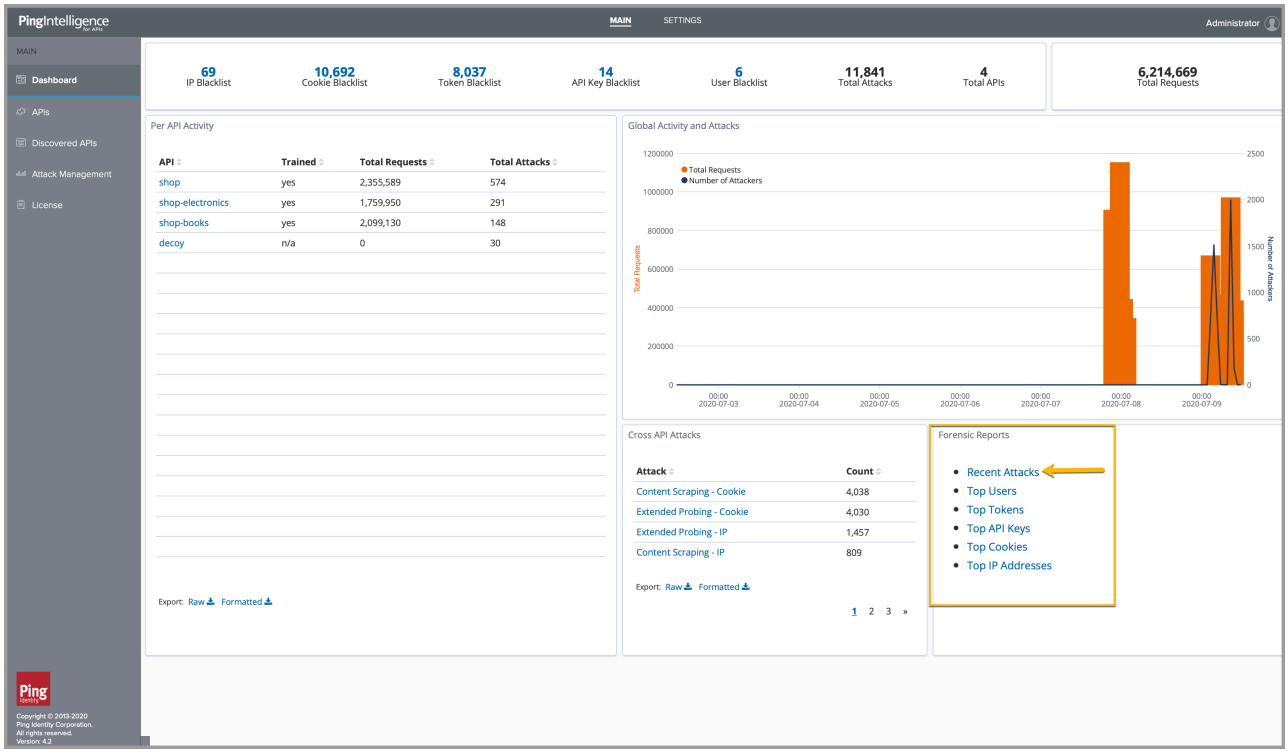
The client activity is reported in the time intervals of 10-minutes.

View the Client Forensic report

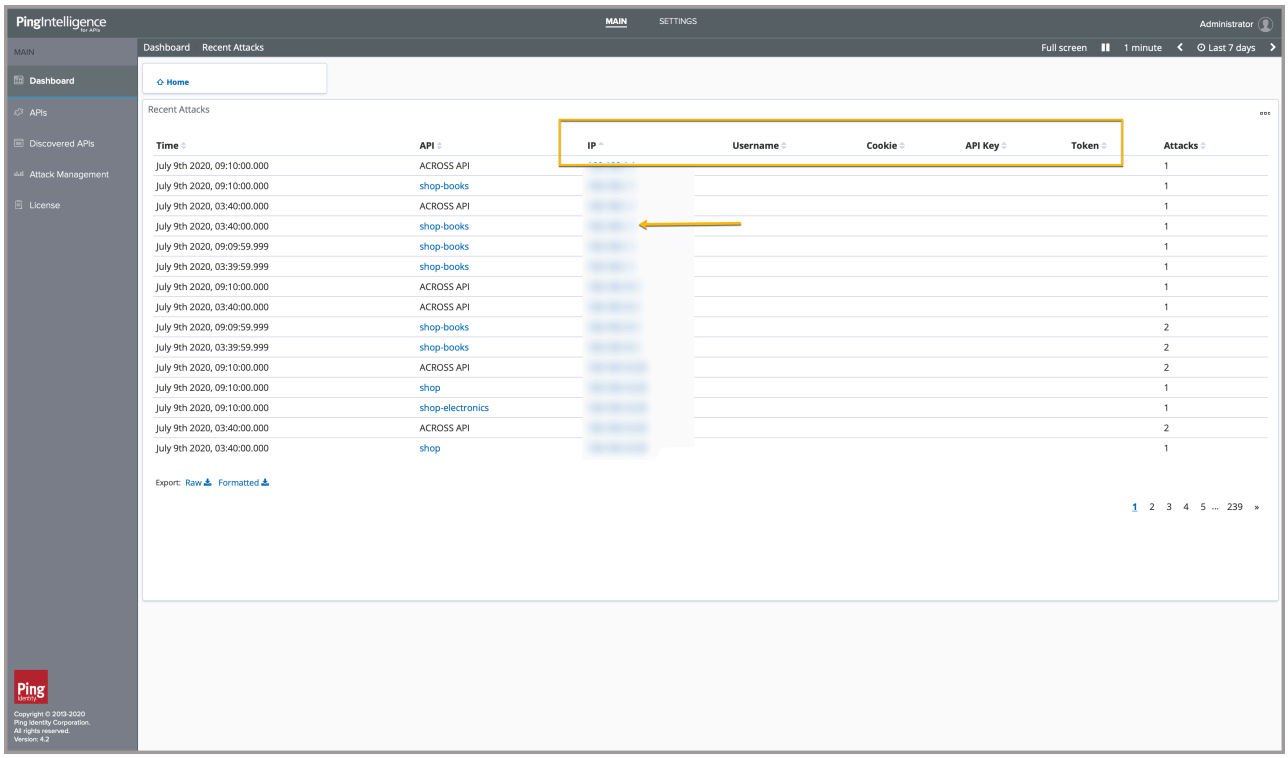
To know the details of client activity before an attack, complete the following steps :

ⓘ **Note:** The steps are explained using IP Address as the client identifier. You can follow the same process to retrieve client forensics for other client identifier types.

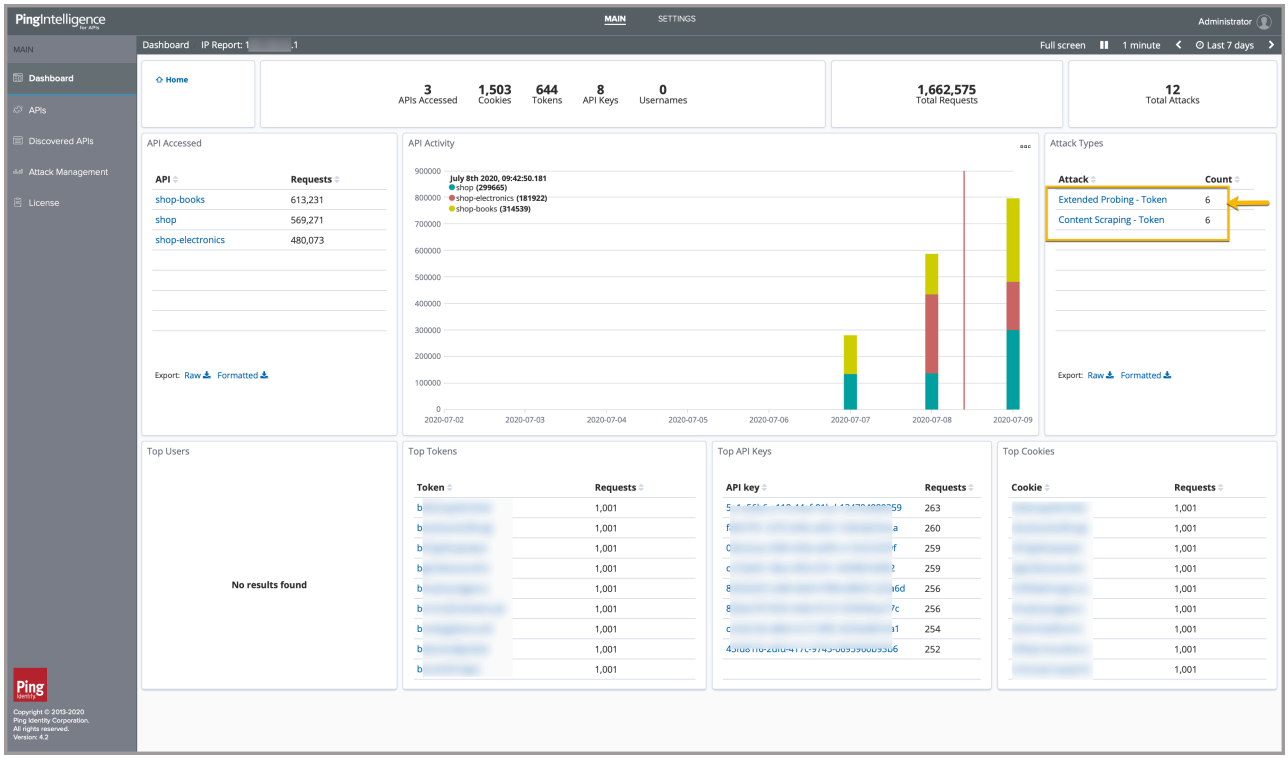
1. Click **Recent Attacks** in **Forensic Reports**, to open the Recent Attacks report.



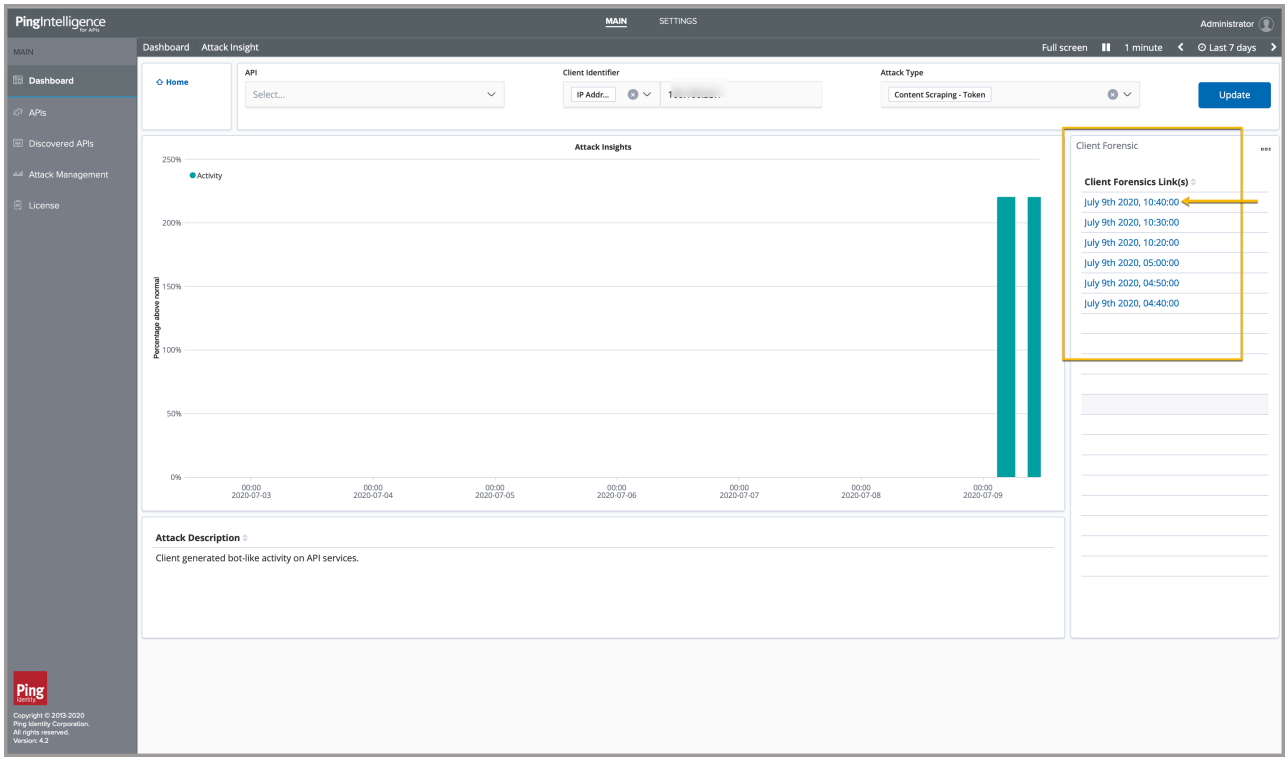
2. Click on  icon next to **IP** to sort the recent attacks for that client identifier type. Click on the IP Address for which the client forensics are to be retrieved. This opens the detailed report for the client.



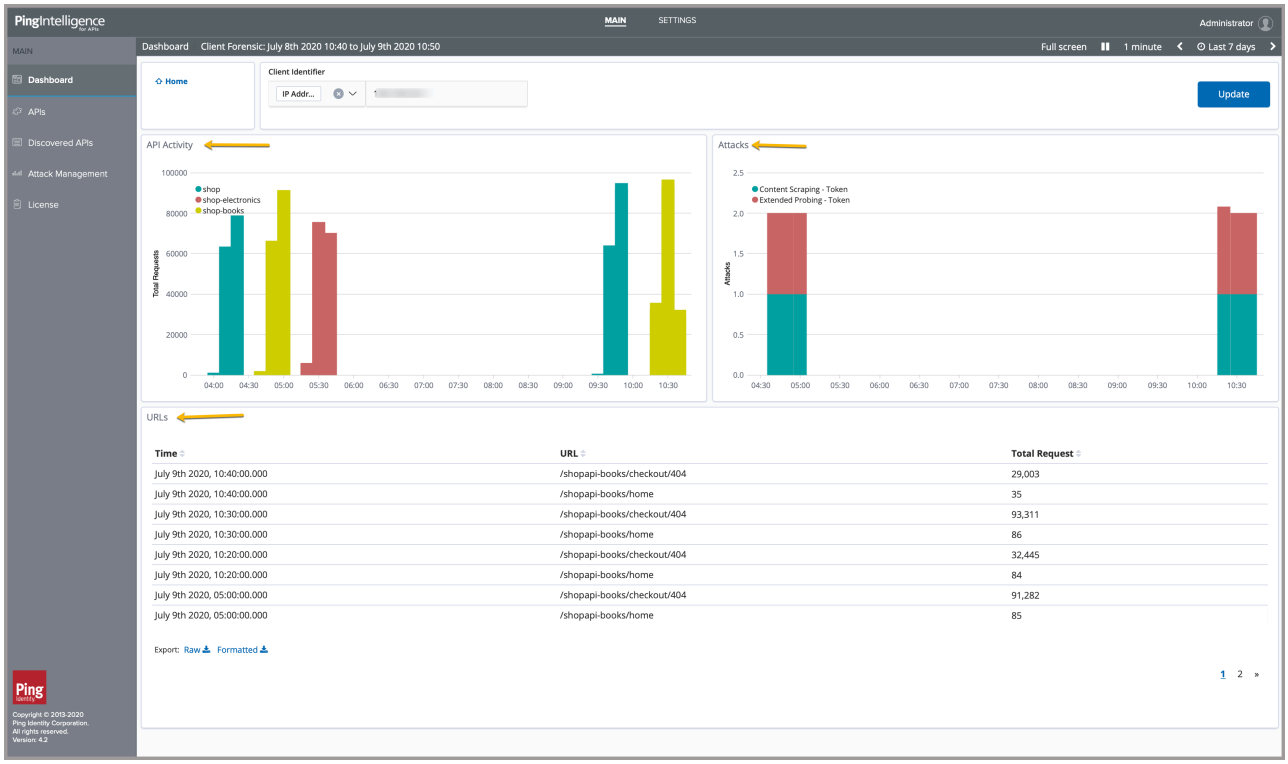
3. In the IP report, select the **Attack** for which the client forensics are required. This opens the Attack Insights report for the client.



4. In the Attack Insight report, select the **Attack Time** from Client Forensics Links to open the Client Forensics report for the client.



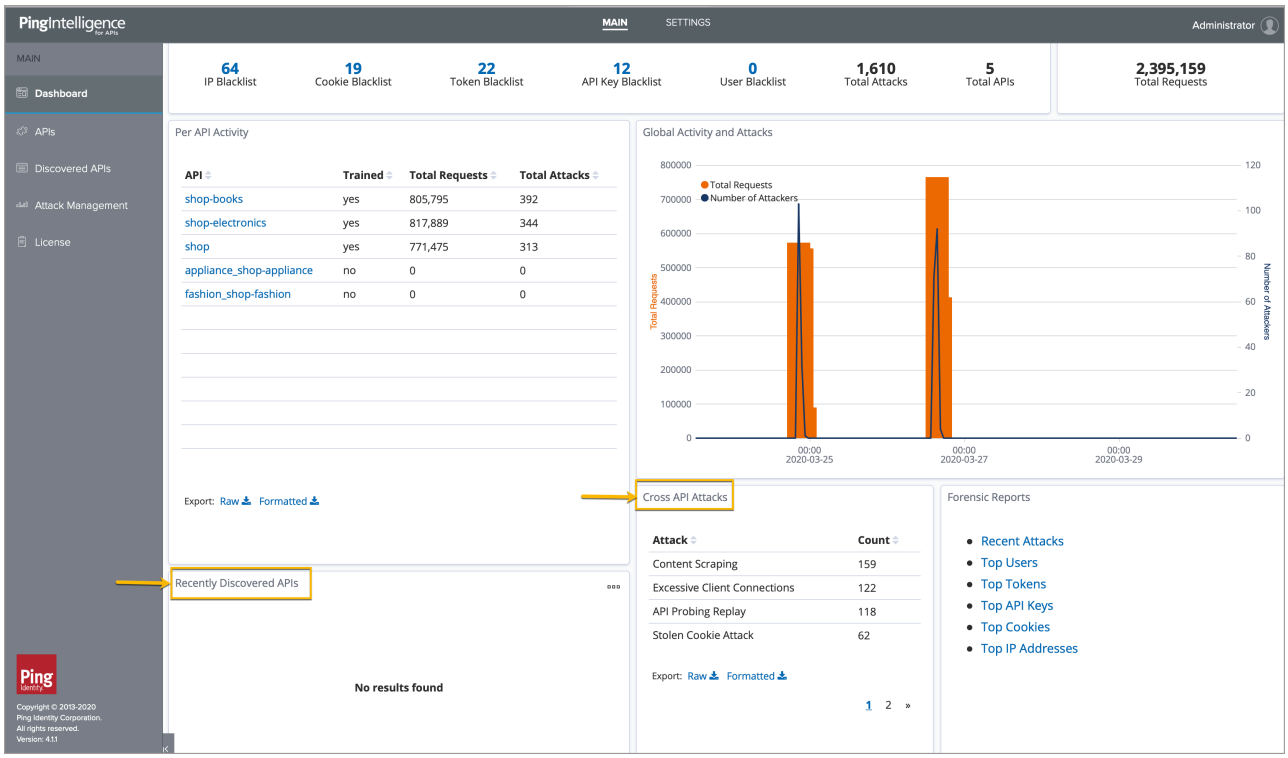
5. The Client Forensic report provides detailed client activity prior to the attack time that is selected in the step-4. It displays details like the APIs and URLs accessed by the client, other attacks executed by the client. It also provides the count of such requests and attacks.



> ⓘ **Note:** Changes to the **Time Range** filter on the top-right corner of the Dashboard will not impact the results retrieved by the Client Forensic report.

Cross API attacks and recently discovered APIs

PingIntelligence for APIs Dashboard provides data on attacks on the APIs in your ecosystem, along with their count. PingIntelligence for APIs can detect per API attacks and across API attacks. For more information, see *REST API attack types* on page 331.

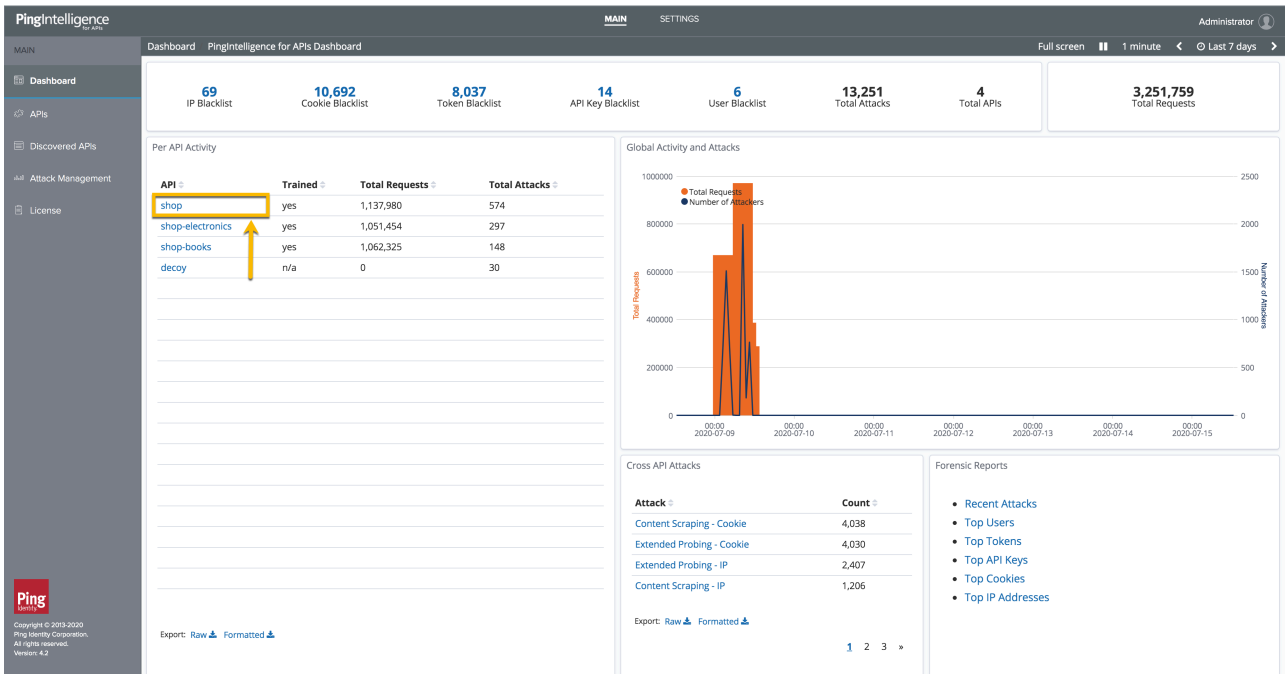


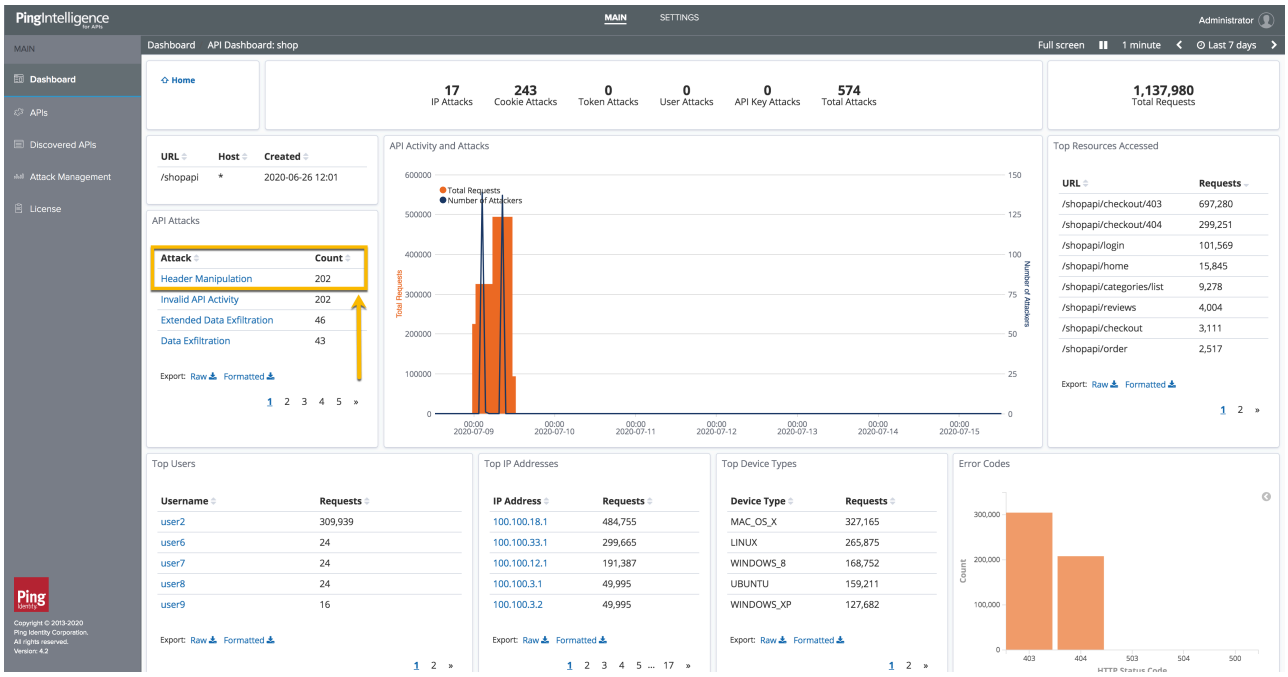
The Dashboard also displays APIs discovered in your ecosystem. For more information, see *Discovered APIs* on page 471.

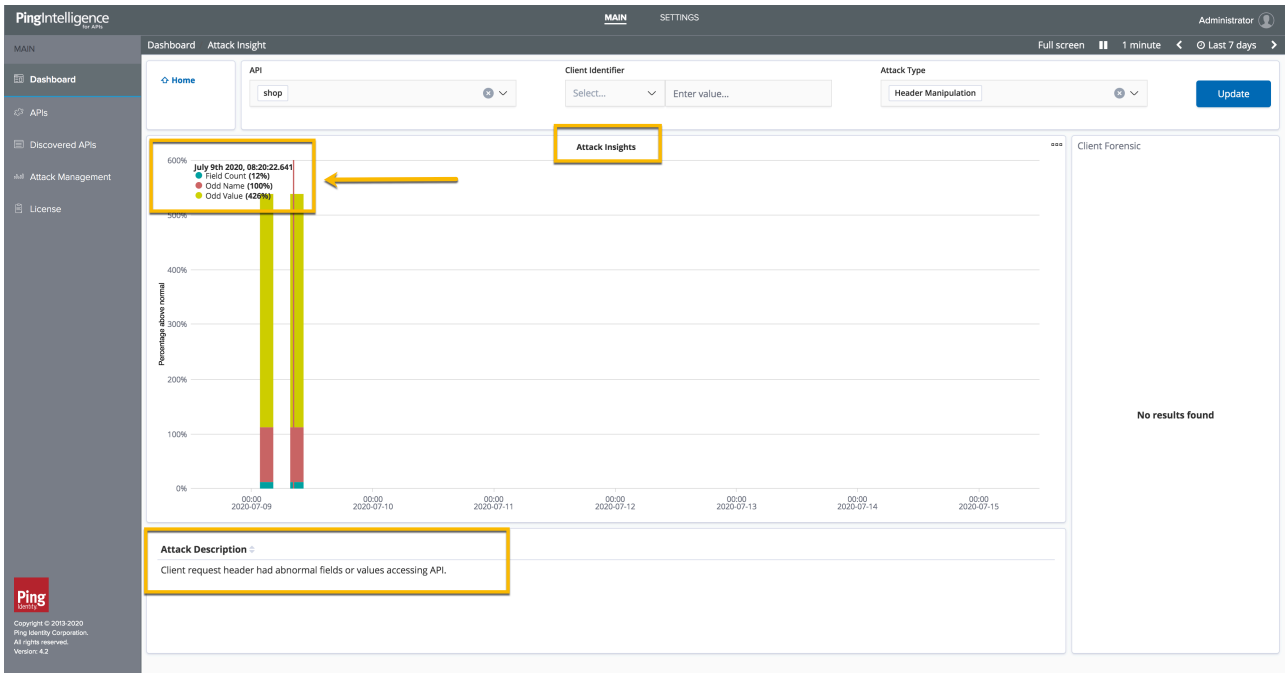
Attack insights

Attack insights provides information on why an attack was identified. The Dashboard displays the percentage by which the normal behavior was exceeded and hence and attack was reported. The attack insight is rendered for a specific API, a client identifier and an attack type, for example, header manipulation attack. To view the attack insight, navigate to the main **Dashboard**. Click on API name in **Per API Activity**. In the API Dashboard page that is displayed, click on the attack type to display the Attack Insight bar charts for that specific attack. To view the percentage deviation from the normal behavior, hover your mouse over the bar chart.

In the following screenshot, the deviations are 12, 100, and 426% from the normal behavior.







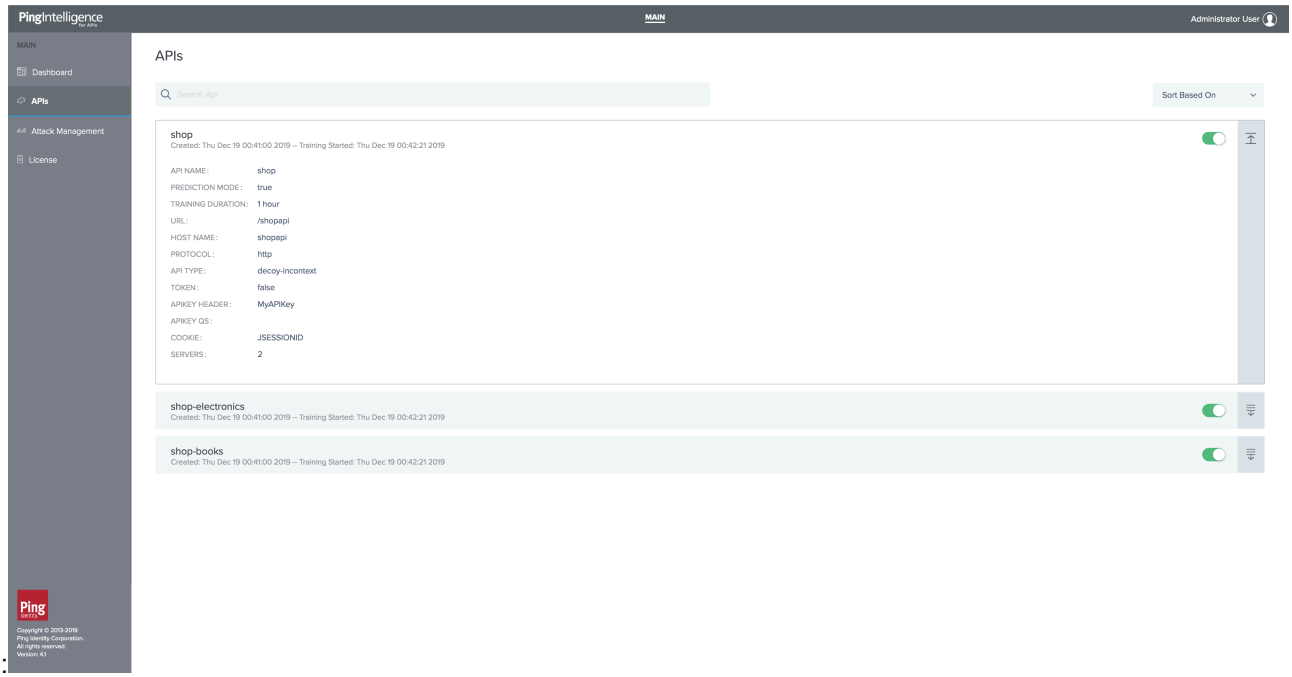
APIs

The API tab displays all the APIs available in ABS AI engine and displays the following information:

- **API name**: API name used by PingIntelligence
- **Prediction mode**: A `true` status means that at least one system generated threshold value is set, while a `false` status means that the API is still under training mode
- **Training duration**: The minimum configured time in hours configured in ABS AI engine to train an API. This is configured in `abs_init.js` in ABS. For more information, see *AI Engine training* on page 312
- **URL**: API basepath URL configured in the API JSON file. For more information, see *API JSON definition*
- **Host name**: Host name of the API configured in the API JSON file. For more information, see *API JSON definition*
- **Protocol**: The protocol configured in the API JSON file. For more information, see *API JSON definition*
- **API type**: API type can be `regular`, `decoy - incontext`, or `decoy-out-of-context`. For more information on deception, see *API deception environment* on page 218
- **Token**: A `true` status means that PingIntelligence will use OAuth tokens for reporting and attack detection. For more information, see *API JSON definition*
- **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 *API JSON definition*
- **Cookie**: The cookie value configured in the API JSON file and used for reporting and attack detection. Displays blank, if cookie was not configured in API JSON. For more information, see *API JSON definition*
- **Servers**: The backend API server configured in the API JSON file - "*" supports all the host names. For more information, see *API JSON definition*

Using the toggle button, you can 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 keeps fetching API data from ABS AI engine. 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. Here is a screenshot of APIs tab:

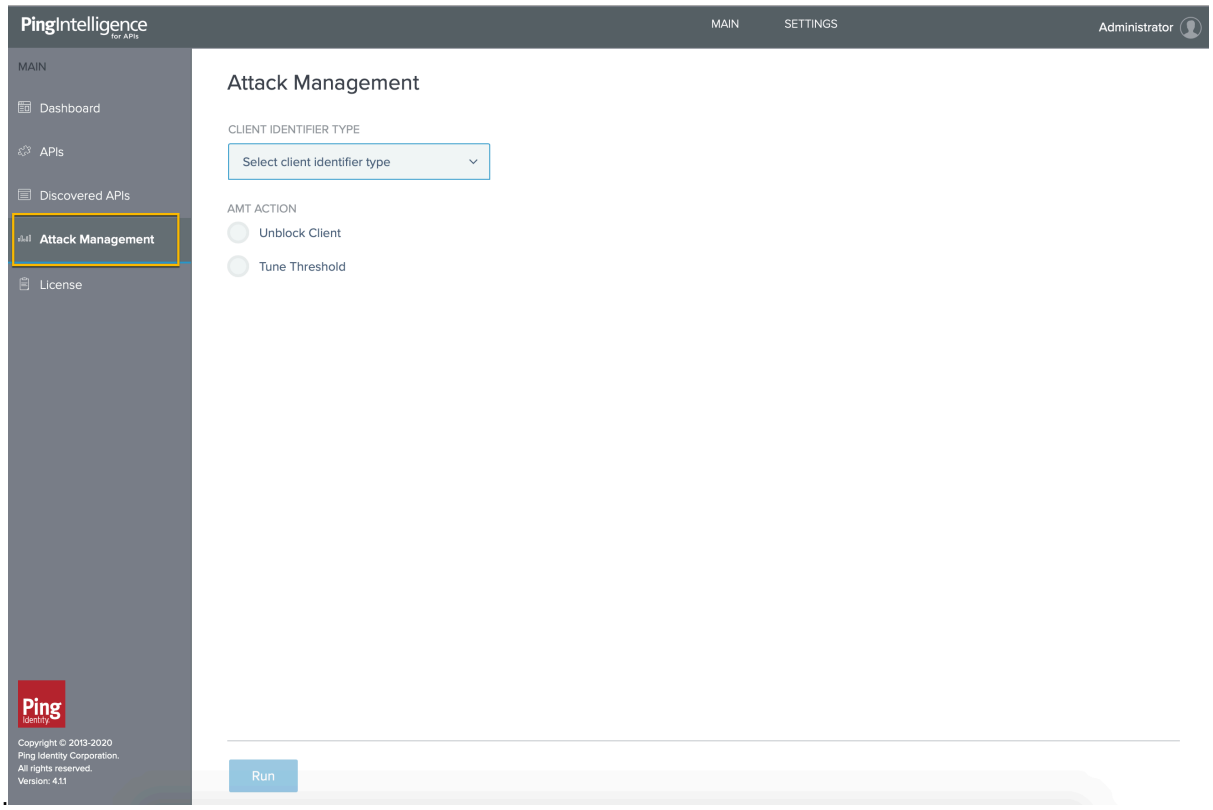


Attack management

The attack management feature of PingIntelligence for APIs Dashboard supports unblocking of clients and tuning thresholds values for attacks. Click on the **Attack Management** tab on the left pane to access it.

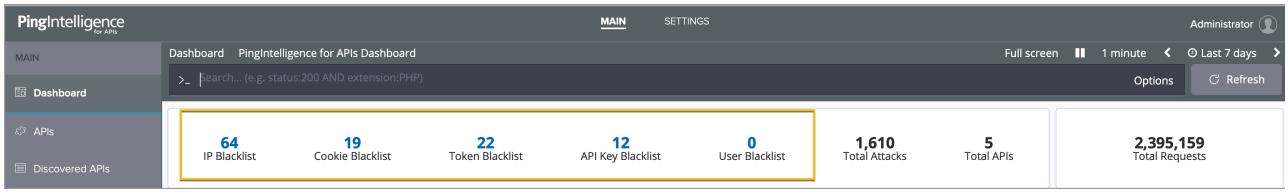
ⓘ **Note:** The Attack management feature is available only for an Admin user. You need to have Admin user privileges to perform **Unblock** and **Tune** operations on a client identifier.

The following screenshot illustrates the Attack Management UI.



Interactive blacklists

The PingIntelligence for APIs Dashboard provides the capability of unblocking or tuning a blacklist directly from the Dashboard. The user can select the client identifier and the Attack management action from the Dashboard. For more information, see *Interactive blacklists* on page 22. The following screen shot shows the client identifier blacklists across APIs in the Dashboard.



> ⓘ **Note:** When the user initiates Attack management from the Dashboard, the values for the client identifiers are auto-populated except the API key key-name.

Unblock a client identifier

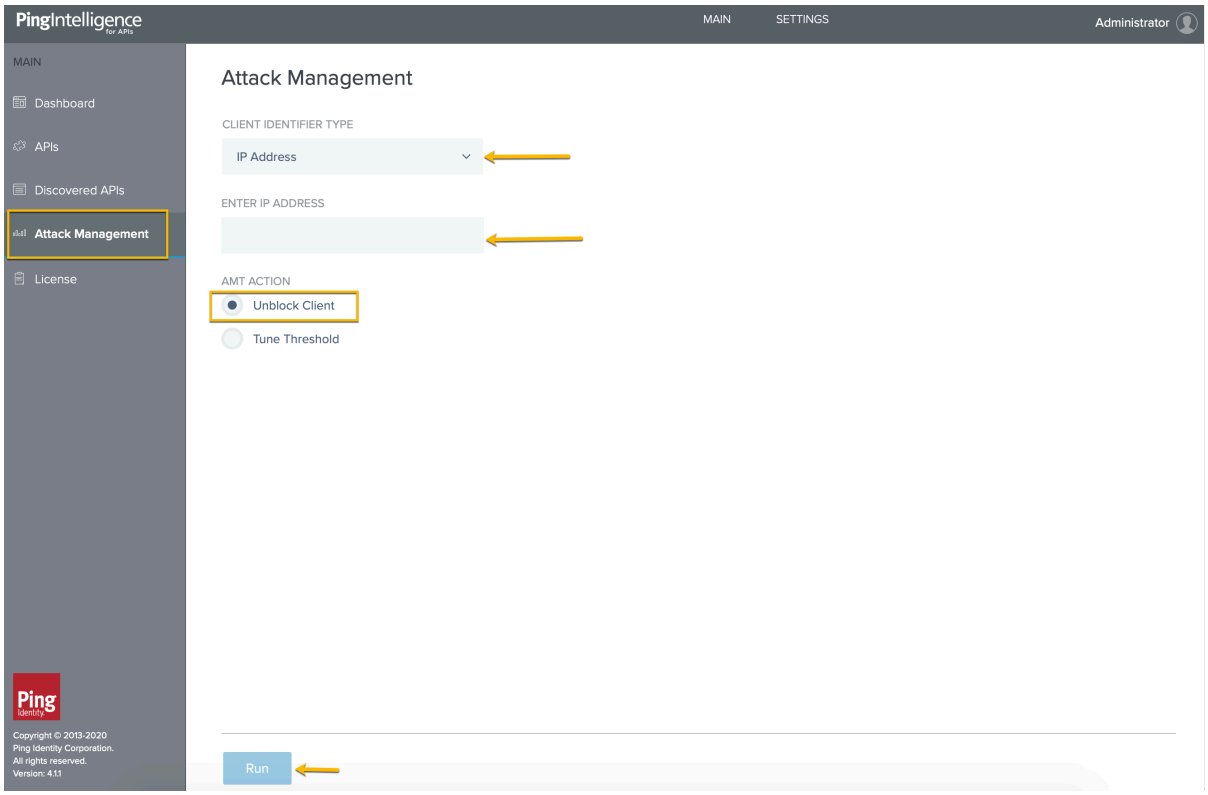
Complete the following steps to unblock a client identifier:

1. Select the type of client identifier from the **Client Identifier Type** list.
2. Enter the value of the client identifier.

> ⓘ **Note:** For API Key and Cookie, enter the name and the value.

3. Select the **Unblock Client** check box.
4. Click **Run**.

The following screen shot illustrates the unblock client operation.



The unblock operation deletes the client identifier from the PingIntelligence ASE and ABS AI engine blacklist. To verify that the client identifier has been deleted from ASE, run the **`view_blacklist`** CLI command or blacklist REST API in ASE. To verify that the client identifier has been deleted from ABS, use the `attacklist` REST API. For more information on ABS blacklist, see *ABS blacklist reporting* on page 342.

ⓘ **Note:** The API keys will not be deleted from the blacklist immediately in ASE if the API Key key-name is not entered. The deletion is delayed until ASE retrieves the blacklist data from ABS.

Tune threshold

To address false positives, the **Attack Management** feature supports automatic threshold tuning. When tuning thresholds for a specific client identifier, the Attack management functionality does the following:

1. It fetches all the attacks flagged for the client identifier from ABS AI Engine.
2. After it has identified all the attacks, it increases the threshold values for those attacks. At this point, the threshold has moved from `system` defined to `user` defined. For more information on thresholds, see *Tune thresholds for false positives* on page 317.

Complete the following steps to tune thresholds:

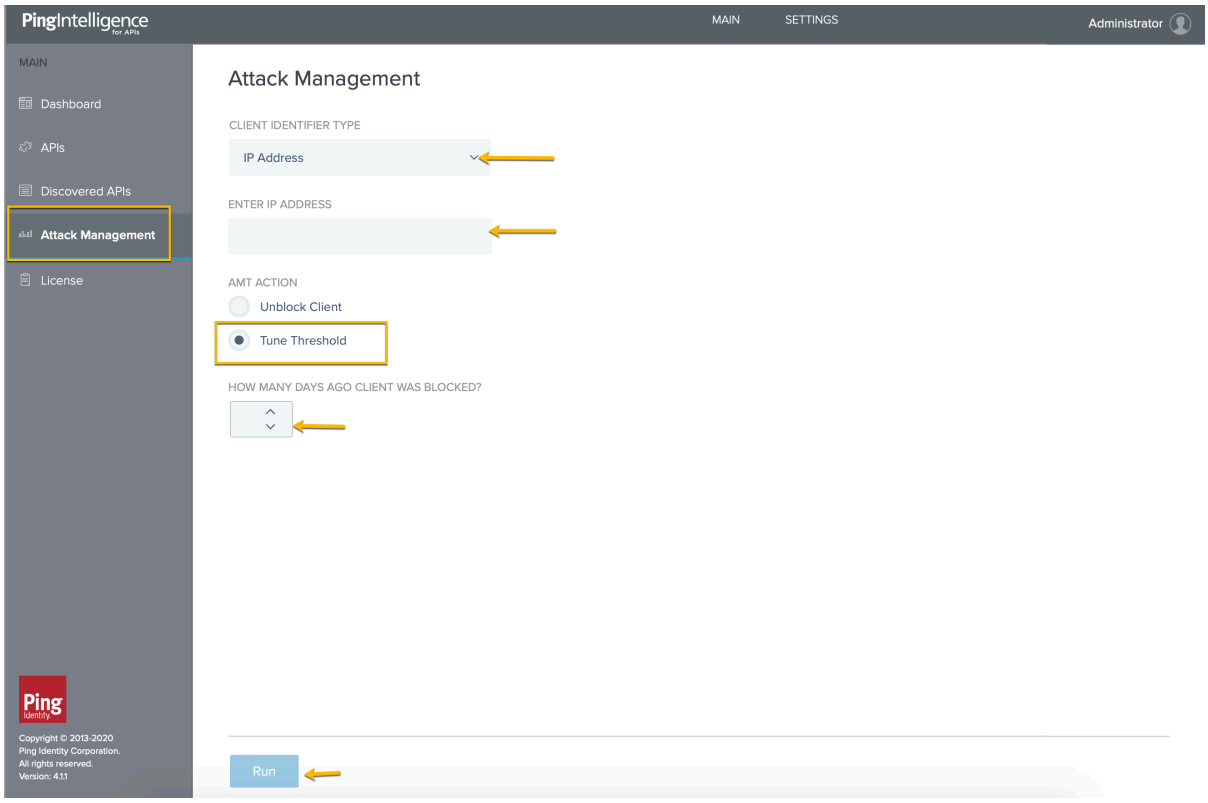
1. Select the type of client identifier from the **Client Identifier Type** list.
2. Enter the value of the client identifier.
3. Select the **Tune Threshold** check box.

4. Provide the approximate number of days since the client was blocked. The maximum value is 30-days.

> ⓘ **Note:** The value for **How many days ago client was blocked?** gets auto-populated when Attack Management is initiated from the Dashboard interactive blacklist. The value is calculated as follows,
>
> ```
> How many days ago client was blocked? = Current date - Attack detection
>  date + 1
> ```
>
> When auto-populating, if the calculated value is more than 30 days, it is trimmed down to 30.You can use the same formula when populating the value manually. The Attack detection date for a client identifier is available in the interactive blacklists.

5. Click **Run**.

The following screen shot illustrates tuning threshold for a client identifier.

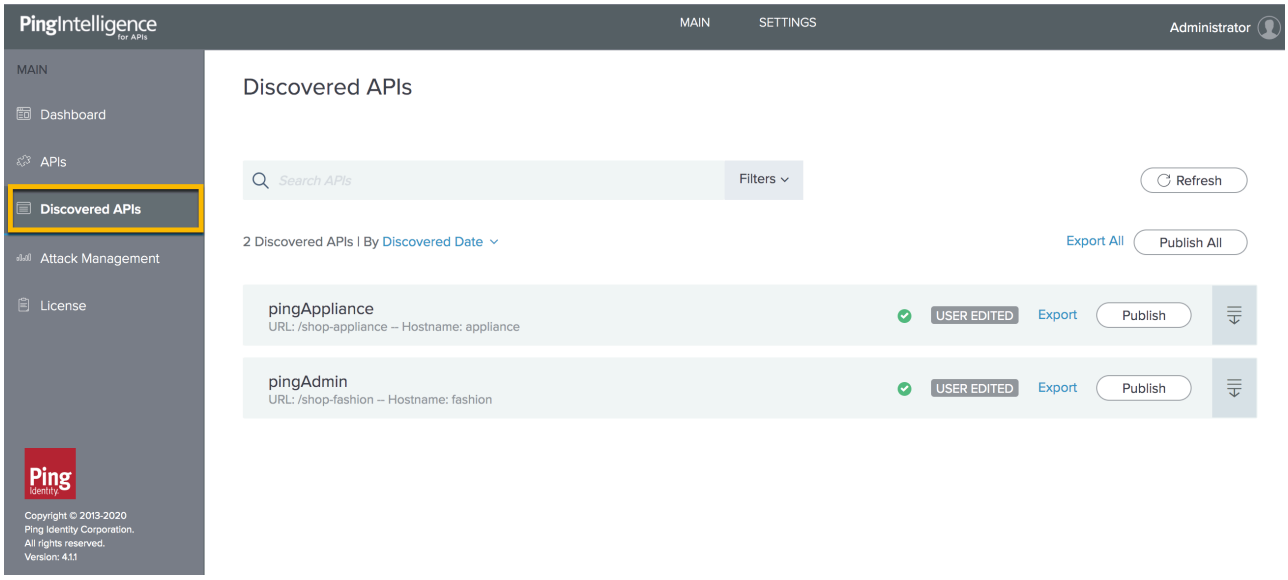


Discovered APIs

API discovery is a process to discover APIs in your API environment. The discovery process involves all PingIntelligence components.

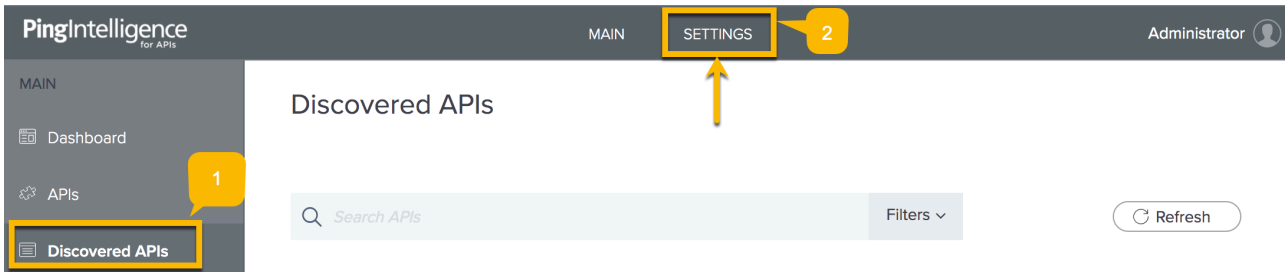
- **ASE** - A `root` API is defined in ASE for the discovery process to start. The `root` API access log data is sent to ABS AI engine for processing.

- **ABS AI engine** - The ASE access logs are processed to discover APIs in your environment.
- **Dashboard** - Displays and manages 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 **Discovered APIs** > **Settings** as shown in the screenshot below.



Discovery settings consists of the following three parts:

- **Mode** - Configure the mode in which APIs are published to 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.

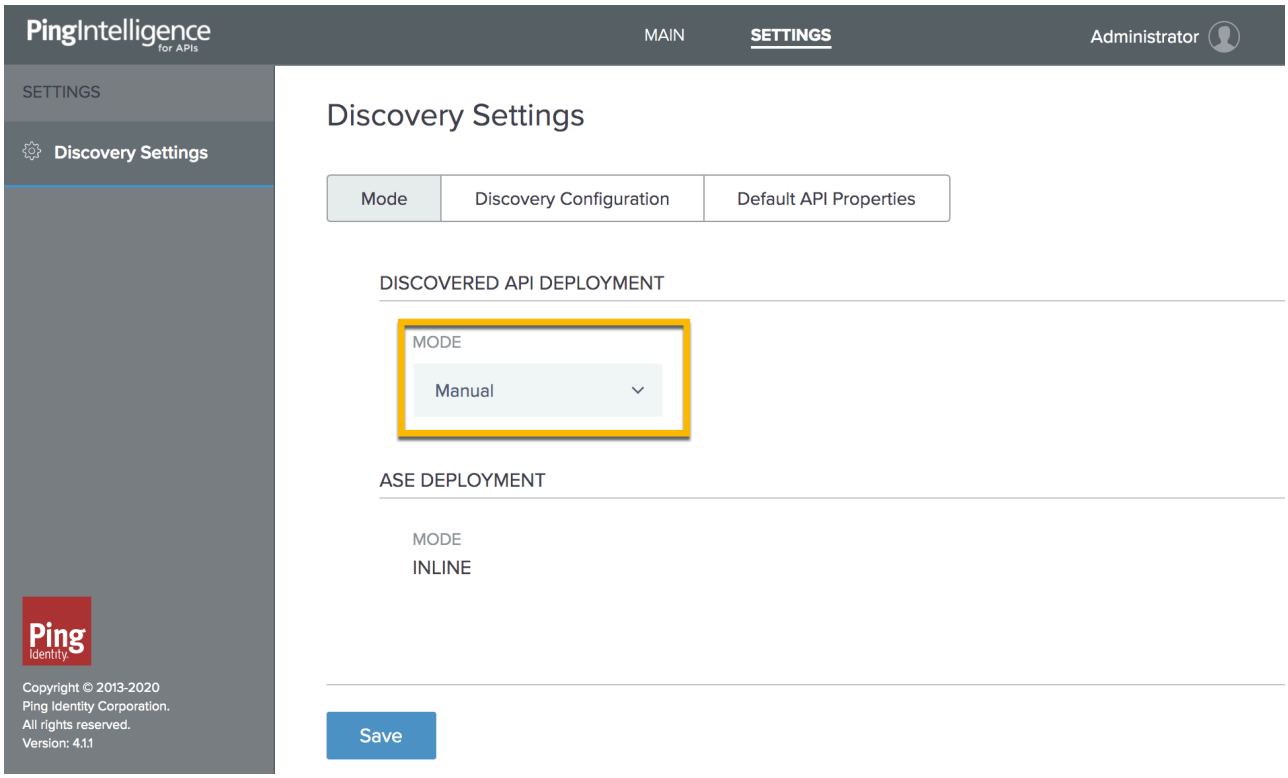
The following sections explain each parts of Discovery settings in detail.

Mode

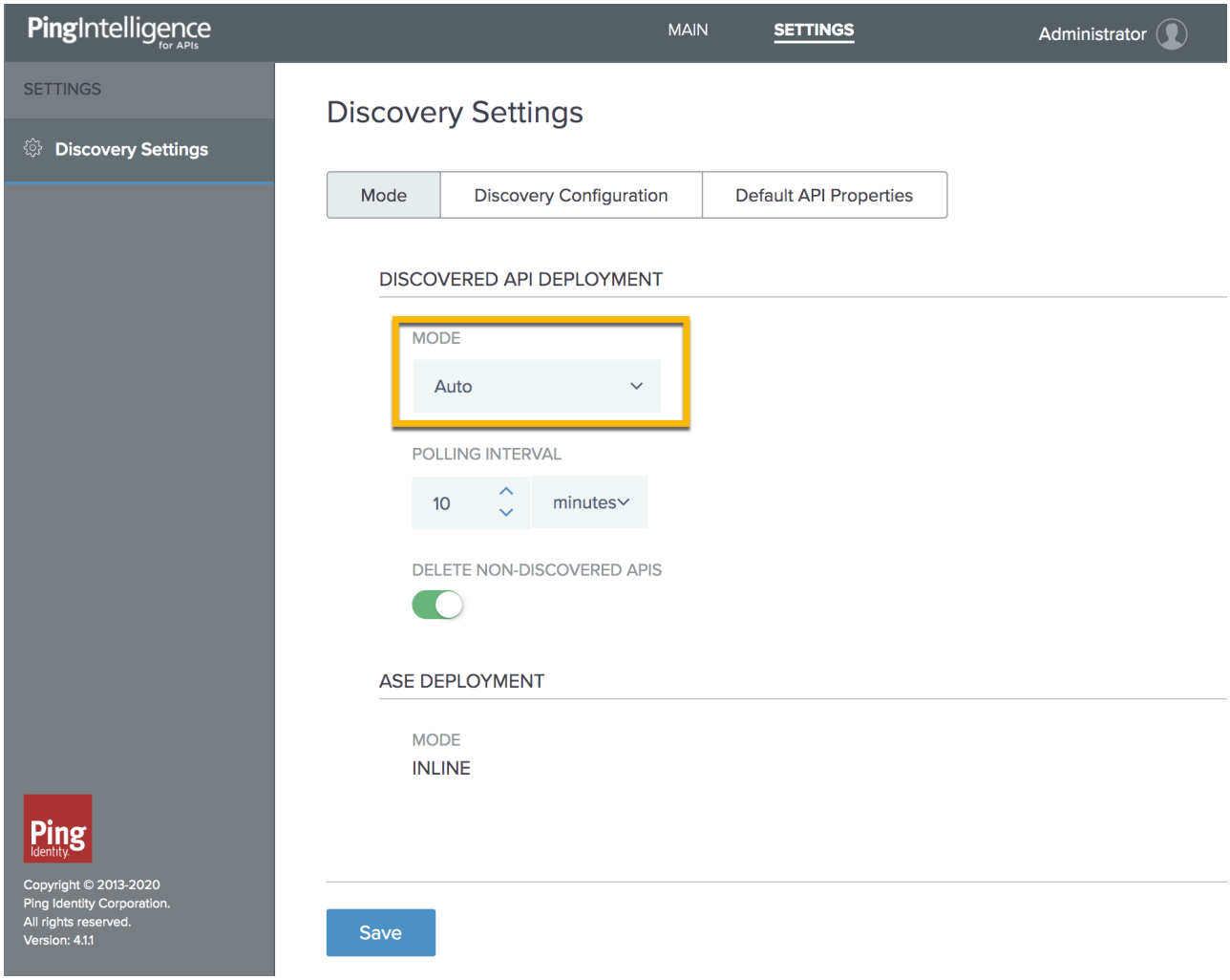
Configure the mode in which Dashboard publishes the discovered APIs to ASE. The two modes are:

- Manual mode - In the manual mode, you can review the discovered APIs, edit the properties of the APIs and then publish one or more APIs to ASE. For more

information on editing the discovered APIs, see *Edit the discovered APIs* on page 477.



- Auto mode - In the auto mode, Dashboard automatically publishes the APIs to ASE after a configured time interval. In auto mode, if you edit an API, it is published to ASE 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.



- **ASE Deployment** - Displays the ASE deployment mode - inline or sideband. The deployment mode is configured in the `/pingidentity/webgui/config/webgui.properties` file. Here is a snippet of the `webgui.properties` file to configure the ASE deployment mode.

```
### ase properties
# ASE management url
pi.webgui.ase.url=https://10.96.2.237:8010
# 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=
```

ⓘ **Note:** Make sure that the ASE mode configured in `webgui.properties` matches the configuration in `pingidentity/ase/config/ase.conf` file in ASE.

Discovery Configuration

Configure enabling or disabling discovery from the **Discovery Configuration** tab by toggling the **AI Engine Discovery** button. Configure the following:

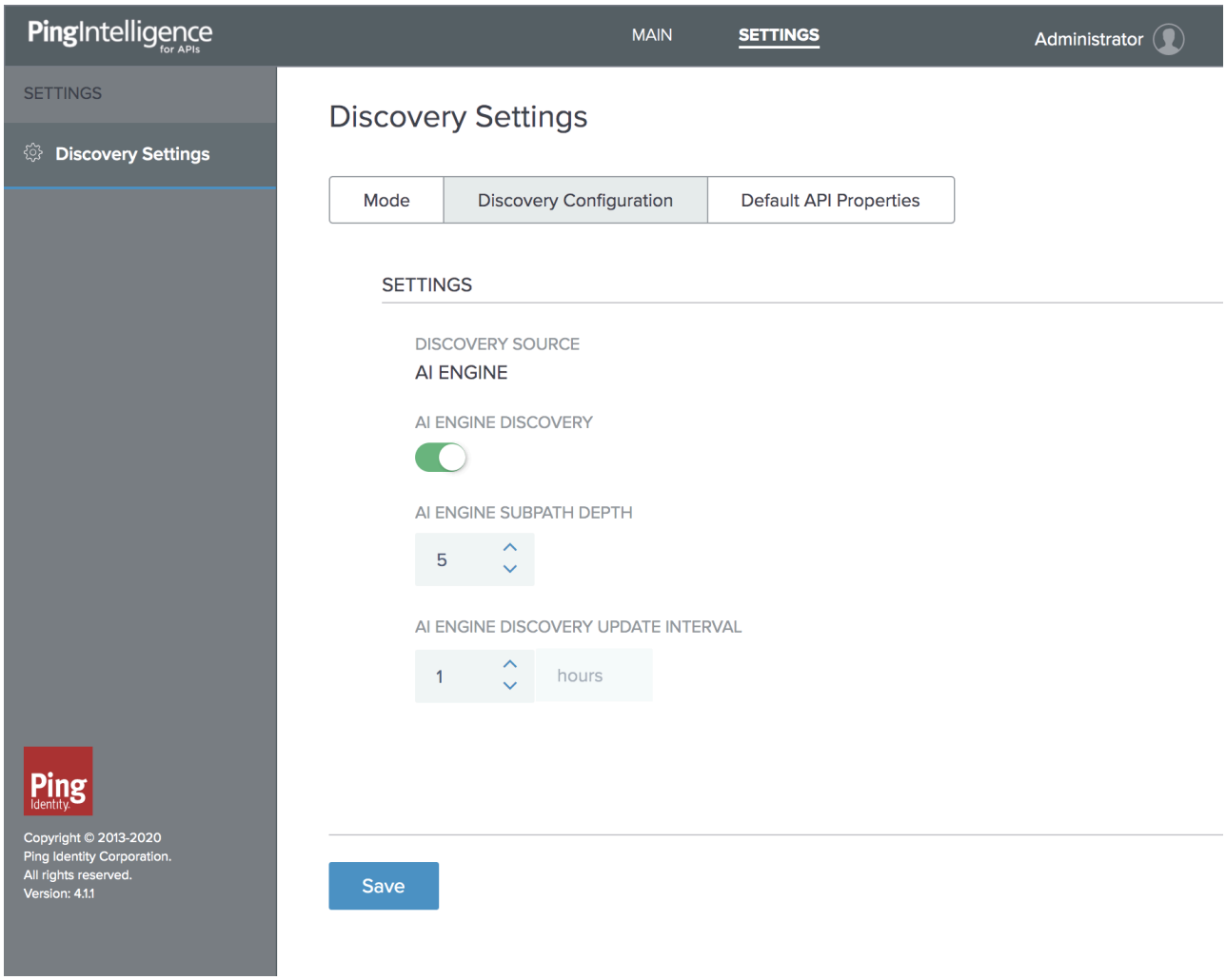
- **Discovery Source** - Dashboard can discover APIs from three sources, ABS AI engine, PingAccess, and Axway API gateway. The discovery source is configured in the `/pingidentity/webgui/config/webgui.properties` file Following is a snippet of the `webgui.properties` file for configuring the discovery source.

  ```
  ### api discovery properties
  # discovery source
  # valid values: abs, axway and pingaccess
  # for axway and pingaccess, see config/discovery.properties
  pi.webgui.discovery.source=abs
  ```

  When the API discovery source is PingAccess or Axway, configure the gateway management URL and credentials in the `/pingidentity/webgui/config/discovery.properties` file. Following is a snippet of the `discovery.properties` file for configuring the credentials.

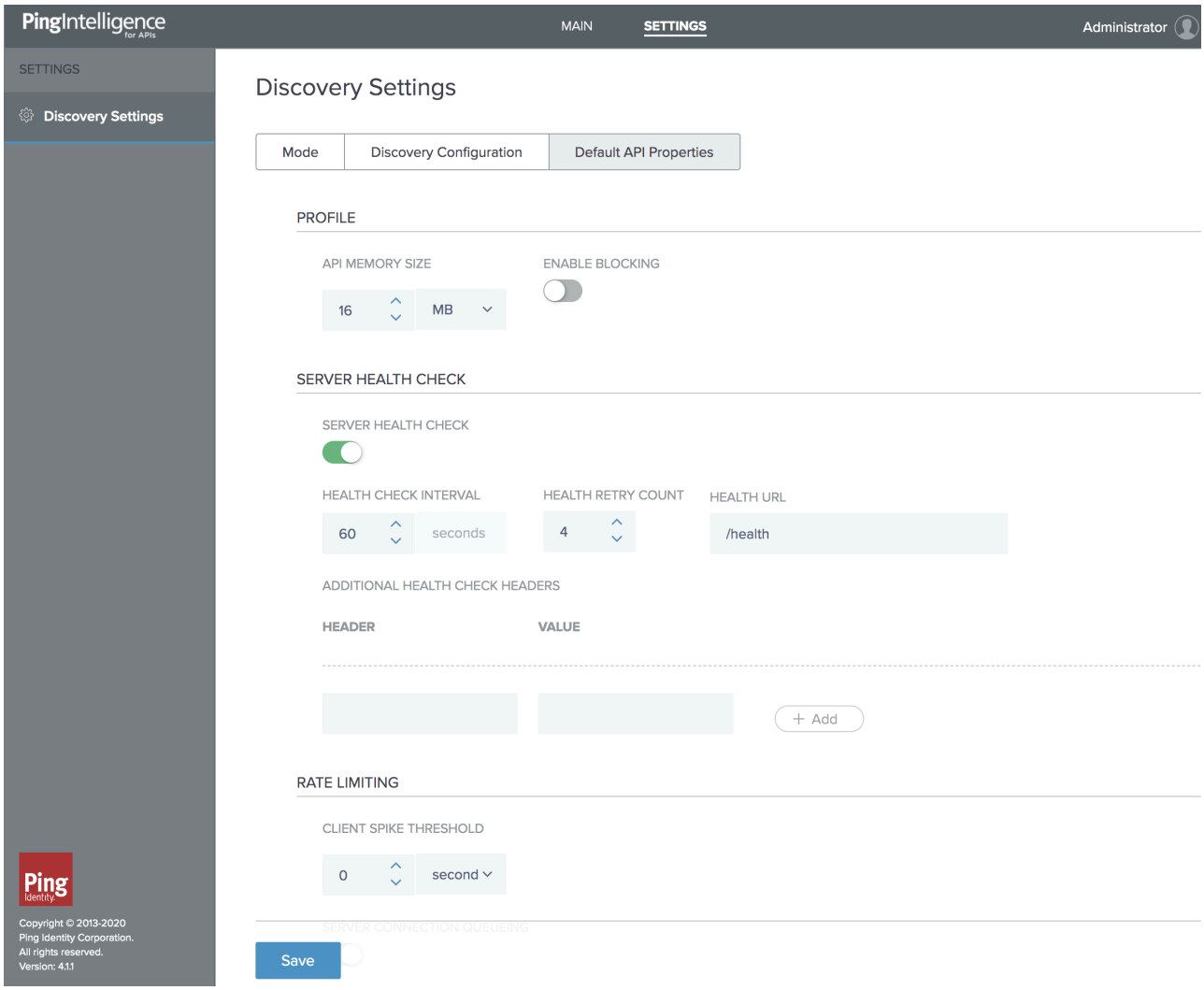
  ```
  ### Axway API Gateway config. Only valid if
   pi.webgui.discovery.source=axway
  # API Manager URL
  axway.management.url=https://127.0.0.1:8075/
  # API Manager admin username
  axway.management.username=username
  # API Manager admin password
  axway.management.password=xxxxxxxx

  ### 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=xxxxxx
  ```

- **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* on page 321.
- **AI Engine Subpath Depth** - Defines the number of subpaths used to uniquely discover the base path of a new API. The maximum value is 6. For more information, see *Discovery Subpaths* on page 325.
- **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

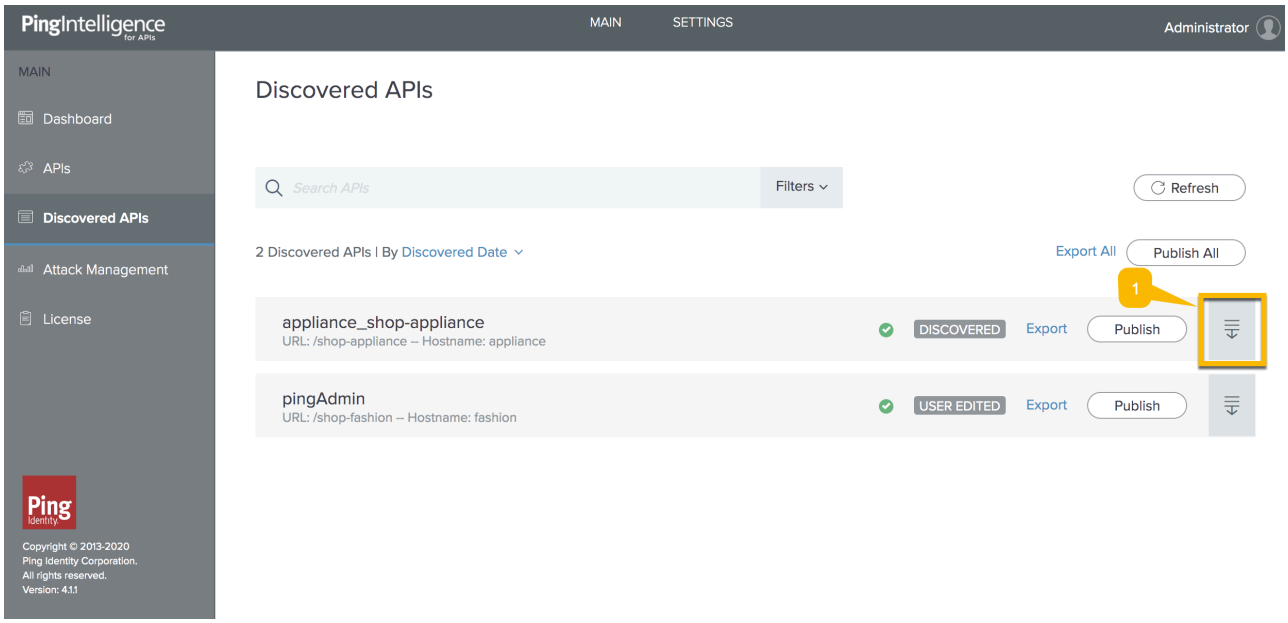
You can configure the default API JSON properties from this tab. These properties apply to all the discovered APIs. You can edit the properties of the discovered APIs in the manual mode before publishing to ASE. For more information on the API properties, see *Define an API JSON*



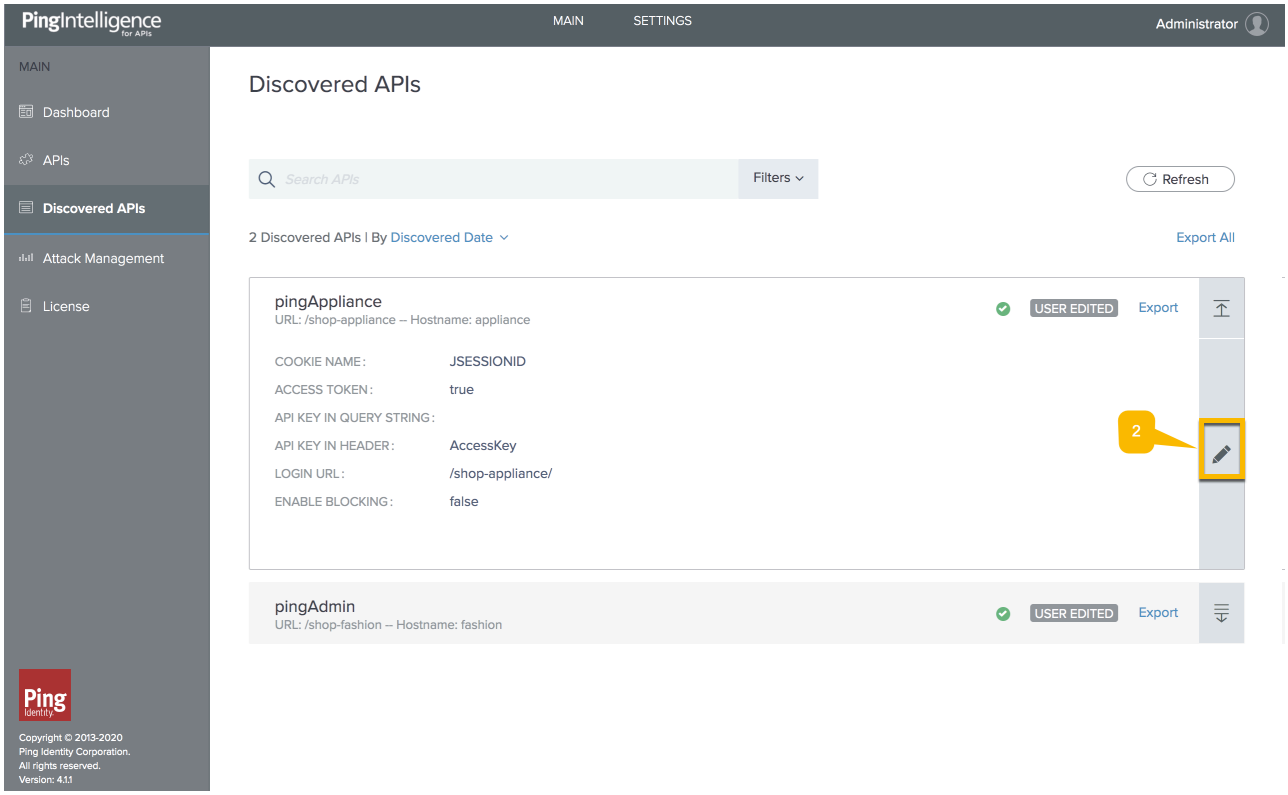
Edit the discovered APIs

You can edit the discovered APIs from the **Discovered APIs** page. To edit an API, click on the buttons as shown in the next two screenshots.

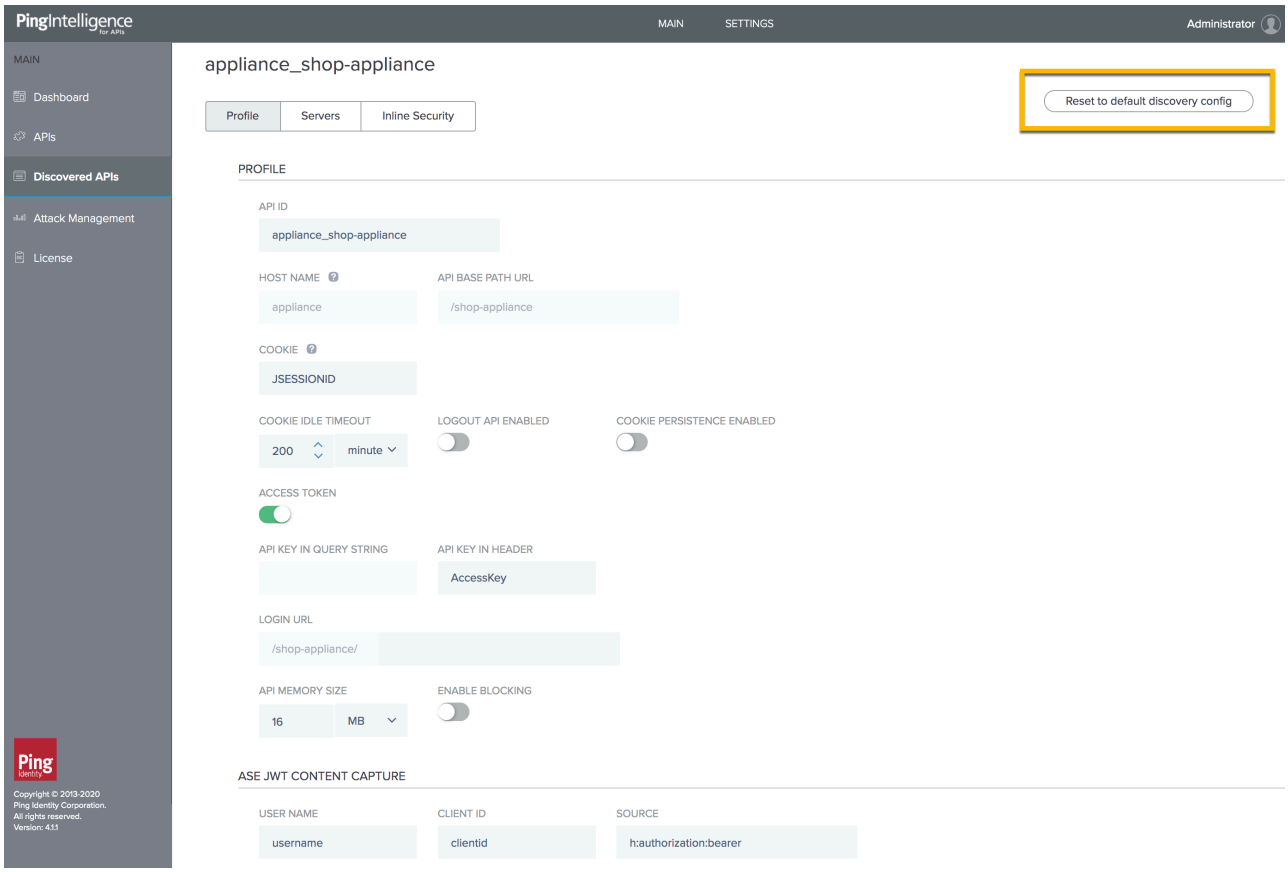
Step 1



Step 2



The edit API page is displayed when you click on the edit button as shown in step 2.



The edit API page is divided into three tabs.

- Profile
- Servers
- Inline Security

These are the same properties that you configure when you define an API JSON in ASE. For more information on defining an API JSON, see *Define an API JSON*. You can also reset the edited changes by clicking on the **Reset to default discovery config** button on the top-right corner. This resets the API properties to the one that was set during the *Configure API discovery* on page 472 step.

Configure 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, stop the dashboard engine, edit the `dashboard.properties` file and then start the dashboard engine. See, *Start and stop Dashboard* on page 106 on how to start and stop each component individually.

```
# 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:

Parameter	Description
ABS	
abs.host	IP address of the ABS server ⓘ **Note:** Two options exist to choose an ABS server: 1) Utilize an existing ABS server. 2) 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` Default value is 8080

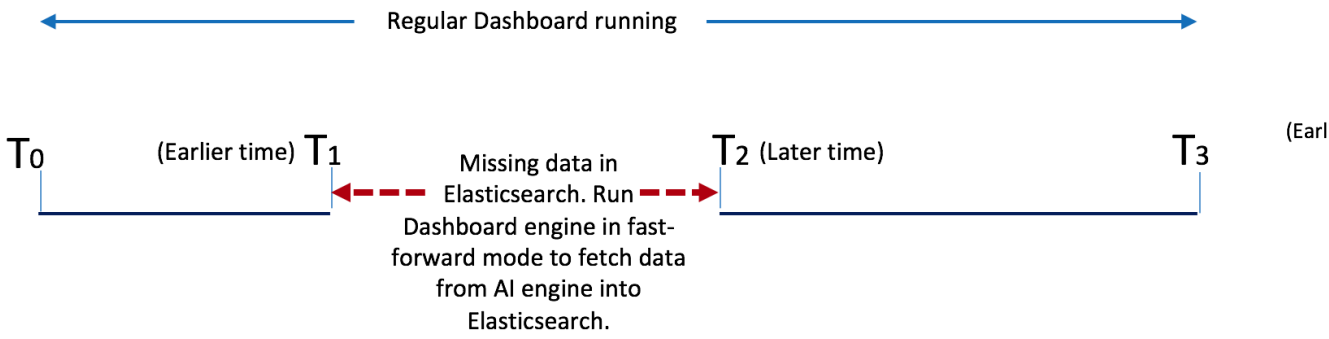
abs.ssl	Setting the value to true ensures SSL communication between ABS and dashboard engine.
abs.restricted_user	When set to `true`, Elasticsearch uses the restricted user header (configured in `pingidentity/abs/mongo/abs_init.js` file) to fetch the obfuscated values of OAuth token, cookie and API keys. 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 header, 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 it to `true` to display PingIntelligence Dashboard. The Dashboard displays attack and metrics data. Set it to `false`, if you do not want to display the Dashboard.
es.url	Elasticsearch URL
es.username	Elasticsearch username
es.password	Elasticsearch password.
kibana.version	Kibana version - default is 6.8.1
dashboard.log.level	Log level for Dashboard Default log level is `INFO`. Another log level is `DEBUG`
Log4j	
publish.log4j2.enable	Set it to `true` to send attack data to syslog server. Set it to `false` to disable sending attack data to syslog server. ⓘ **Note:** Dashboard and Syslog cannot be disabled together.
log4j2.config	The log4j2 config file which logs the attack data.
log4j2.log.level	Log level for log4j. 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 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



Missing data in Elasticsearch

mode:

When you run 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.

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.

ⓘ **Attention:** If your current Dashboard engine is running in `/opt/pingidentity/dashboard/`, make sure that you use a different directory to run 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 Dashboard fast-forward mode.

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 Dashboard queries the AI engine in a single pass.

Parameter	Description
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

Install a new instance of dashboard binary in a different directory in `/opt/pingidentity/`, for example, `/opt/pingidentity/dashboard_fast_forward`. Enter the following command to start Dashboard in fast-forward mode:

```
# /opt/pingidentity/dashboard_fast_forward/bin/start.sh --fast-forward
starting Dashboard Fastforward 4.1
```

Configure dashboard engine for syslog

PingIntelligence dashboard engine supports sending attack information to a `syslog` server. Enable `syslog` support by editing the *`dashboard.properties`* file. By default `syslog` is disabled. Dashboard uses Log4j version2.11.2 to publish attack data to `syslog`.

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/
```

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`.

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`.

syslog.xml

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>
```

Configure server and port number of syslog server

Configure the server and port number of syslog server in `config/syslog.xml` file. Following is a snippet of the `syslog.xml` file displaying the server and port number parameters:

```
 <!--  ### 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"/>
```

attack.log for Splunk

Configure `dashboard.properties` for attack.log

Edit the `pingidentity/dashboard/config/dashboard.properties` file to send the attack data to `attack.log`. By default `syslog` is configured. To send the attack data to `attack.log`, edit the `dashboard.properties` file as shown in the snippet below:

```
### 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/
```

attack_log.xml: 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.

```
<?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>
```

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`.

Dashboard log messages

The following tables list the critical log messages from `dashboard.log` file. 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 a HTTP/(S) protocol error while connecting to 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 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 HTTP status code that is greater than or equal to 300.
Dashboard stopped	This message is logged in `dashboard.log` when Dashboard is shutdown.

Purge dashboard logs

The `purge.sh` script either archives or purges processed access log files which are stored in the `/opt/pingidentity/dashboard/logs` directory.

ⓘ **Note:** When the `purge` script is run, the log files are permanently deleted from the `/opt/pingidentity/dashboard/logs` directory. Always backup the files before deleting.

Located in the `/opt/pingidentity/dashboard/util` directory, the `purge` script deletes logs older than the specified number of days. Run the script using the Dashboard command line.

> ⓘ **Note:** The number of days specified should be between 1-365 days.

For example.

```
/opt/pingidentity/dashboard/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/dashboard/util/purge.sh -d 3
This will delete the data in /opt/pingidentity/dashboard/logs which is older
 than 3 days.
Are you sure (yes/no): yes
removing /opt/pingidentity/dashboard/logs/dashboard.log.2019-02-07 : last
 changed at Sat Feb  9 00:29:43 EST 2019
removing /opt/pingidentity/dashboard/logs/dashboard.log.2019-02-09 : last
 changed at Mon Feb 11 00:29:48 EST 2019
removing /opt/pingidentity/dashboard/logs/dashboard.log.2019-02-08 : last
 changed at Sun Feb 10 00:29:56 EST 2019
Done.
```

Force delete: You can force delete the Dashboard log files by using the `-f` option with the `purge.sh` script. When using this option, the script does not check for confirmation to purge the log files. 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.

```
/opt/pingidentity/dashboard/util/purge.sh -d 3 -f
removing /opt/pingidentity/dashboard/logs/dashboard.log.2019-02-07 : last
 changed at Sat Feb  9 00:31:26 EST 2019
removing /opt/pingidentity/dashboard/logs/dashboard.log.2019-02-09 : last
 changed at Mon Feb 11 00:31:30 EST 2019
removing /opt/pingidentity/dashboard/logs/dashboard.log.2019-02-08 : last
 changed at Sun Feb 10 00:31:35 EST 2019
Done.
```

In the above example, the script force purges the Dashboard log files while keeping log files of 3-days.

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/dashboard/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 data from Elasticsearch

To manage storage on the Dashboard server, you can either archive or purge Elasticsearch data. 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 take a backup of Elasticsearch documents before proceeding with the purge.

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
```

In the following example, **`purge_elasticsearch.sh`** deletes all files older than 3 days. Here 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
```

To delete all data and Elasticsearch templates, use the following:

```
curl -s https://<elasticsearch_ip_address>:<port>/_all -X DELETE -u elastic
```

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.

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

.
.
```

ⓘ **Note:** It is recommended to run purge_elasticsearch.sh during lean API traffic periods.

Purge Web GUI logs

The `purge.sh` script either archives or purges processed access log files and admin log files which are stored in the `/opt/pingidentity/webgui/logs/access/` and `/opt/pingidentity/webgui/logs/admin/` directories respectively.

ⓘ **Note:** When the `purge` script is run, the log files are permanently deleted. Hence it is recommended to always backup the files before deleting.

Located in the `/opt/pingidentity/webgui/util` directory, the `purge` script deletes logs older than the specified number of days. Run the script using the webgui command line.

> ⓘ **Note:** The number of days specified should be between 1-365 days.

For 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
```

Force delete: You can force delete the webgui log files by using the `-f` option with the `purge.sh` script. When using this option, the script does not check for confirmation to purge the log files. 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.

```
/opt/pingidentity/webgui/util/purge.sh -d 2 -f
```

In the above example, the script force purges the webgui log files while keeping log files of 2-days.

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/webgui/util/purge.sh -d 2 -l /backup/
```

In the above example, log files older than `2-days` are archived to the `backup` directory. To automate log archival, add the script to a `cron` job.

API Gateway integration

Akana API gateway integration

Akana API gateway sideband integration

This integration guide discusses the deployment of PingIntelligence for APIs in a sideband configuration with Akana API Gateway. PingIntelligence for APIs in a sideband deployment mode integrates with Akana API Gateway to provide in-depth analytics on API traffic. A PingIntelligence policy is installed in the Policy Manager component of 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* on page 144 .

PingIntelligence for APIs provides JavaScript policy that extracts API metadata from a request and response processed by Akana API Gateway. The API metadata is passed to API Security Enforcer (ASE). Here are a few highlights of the integration solution:

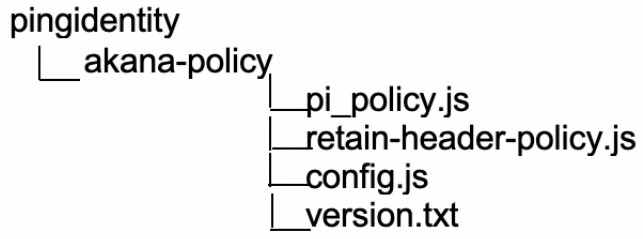
- Support for SSL connectivity through a valid CA signed certificate.

- Support for connection keep alive between Akana gateway and ASE, for faster processing of request and response data.
- 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:** Akana Gateway does not support self-signed certificates.

PingIntelligence policies:

Three policies are made available to support the integration. They are packaged in **pi-api-akana-policy-4.x.x.tar.gz** file. The following diagram shows the directory structure for reference.



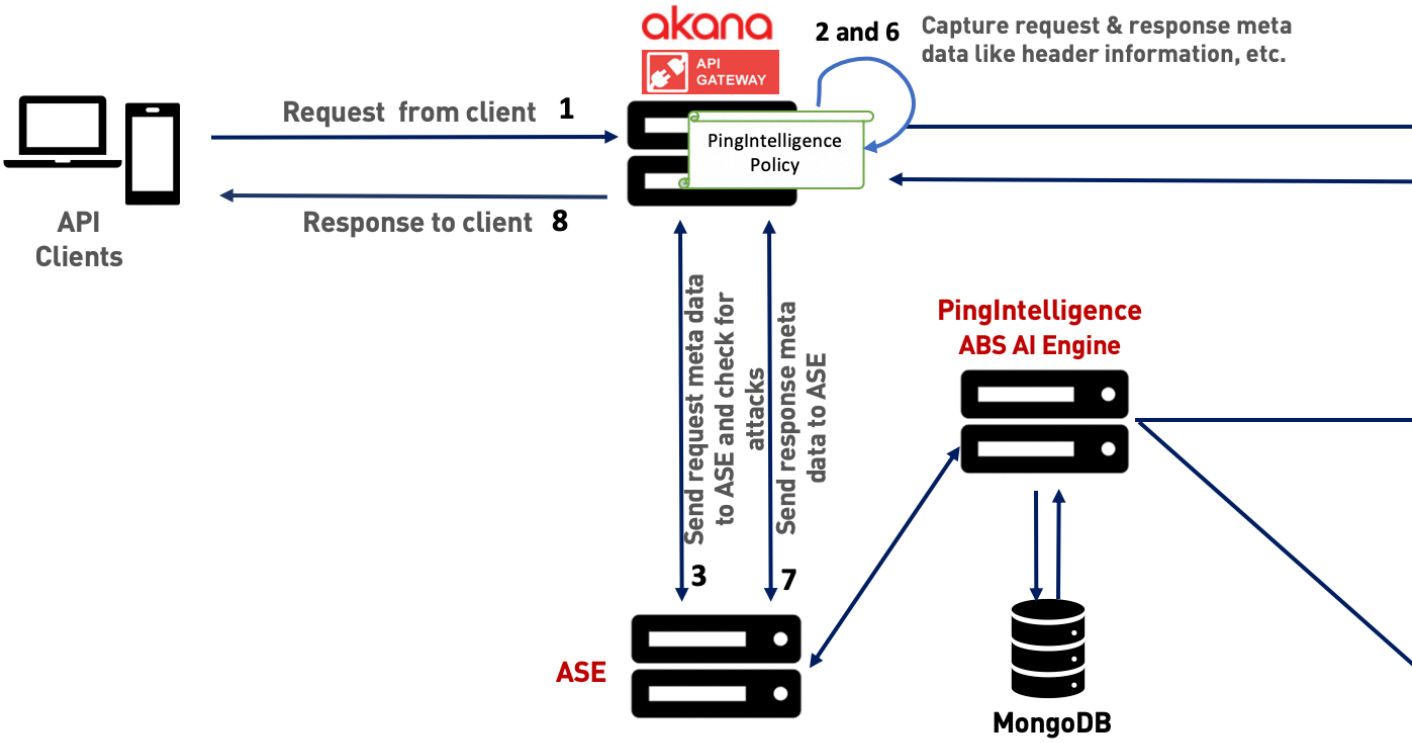
pi_policy.js: This is the main PingIntelligence policy. It extracts the metadata for each API call, formats it into JSON and makes API calls to pass the metadata to ASE.

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 Akana API Gateway:



The traffic flow through the Akana API gateway and PingIntelligence for APIs components is explained below :

1. Client sends an incoming request to Akana API gateway.
2. PingIntelligence policy deployed on 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, tokens against the blacklist. If all checks pass, ASE returns a 200-OK response to the Akana API gateway. If not, a different response code is sent to Akana API gateway (400 or 403). The request information is also logged by ASE and sent to the Ping Intelligence API Behavioral Security (ABS)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 Akana API Gateway.
6. The PingIntelligence policy is again applied on the response to extract the metadata from the server response.
7. 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 API gateway.
8. Akana API gateway sends the response received from the backend server to the client.

Prerequisites

Complete the following prerequisites before deploying PingIntelligence policy on Akana API gateway.

Install PingIntelligence software: PingIntelligence software should be installed and configured. Refer to *Automated deployment* or *Manual deployment*.

Verify that ASE is in sideband mode: Check that 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.

Enable sideband authentication: For a secure communication between 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 *Configure SSL for external APIs* on page 126 for more details.

Generate sideband authentication token: To generate the token in ASE, enter the following command in the ASE command line:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

Enable connection keepalive between gateway and ASE- Navigate to `/opt/pingidentity/ase/config/`. Set the value of enable_sideband_keepalive to `true` in ase.conf file. If the ASE is running stop it, before making the change. Start ASE after setting the value. For more information on ASE configuration, see *ASE configuration - ase.conf* on page 146

Add PingIntelligence ASE APIs

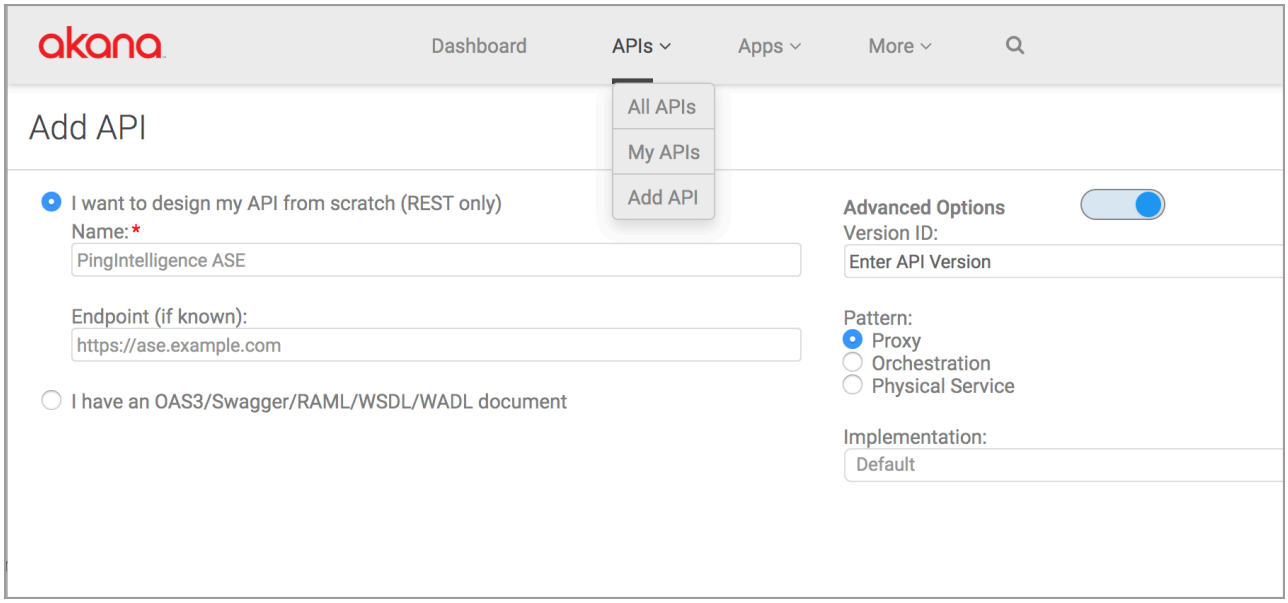
This section explains the steps to add a primary and secondary ASE nodes to Akana API gateway.

> ⓘ **Important:** The primary and secondary ASE APIs should **not be** exposed to external API clients. For more details on securing ASE APIs, see *Secure PingIntelligence ASE APIs* on page 495.

To add ASE APIs to Akana API gateway:

1. Login to **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:
   a. Name of the API in **Name**.
   b. Enter the Endpoint-<http/https>://<ASE-Hostname or IP>/ase.
   c. Enable **Advanced Options**.
   d. Enter API version in **Version ID**.
   e. Select Pattern **Proxy** in the **Pattern** section.
   f. Select an **Implementation**.
   g. Select **Deployment Zones**.

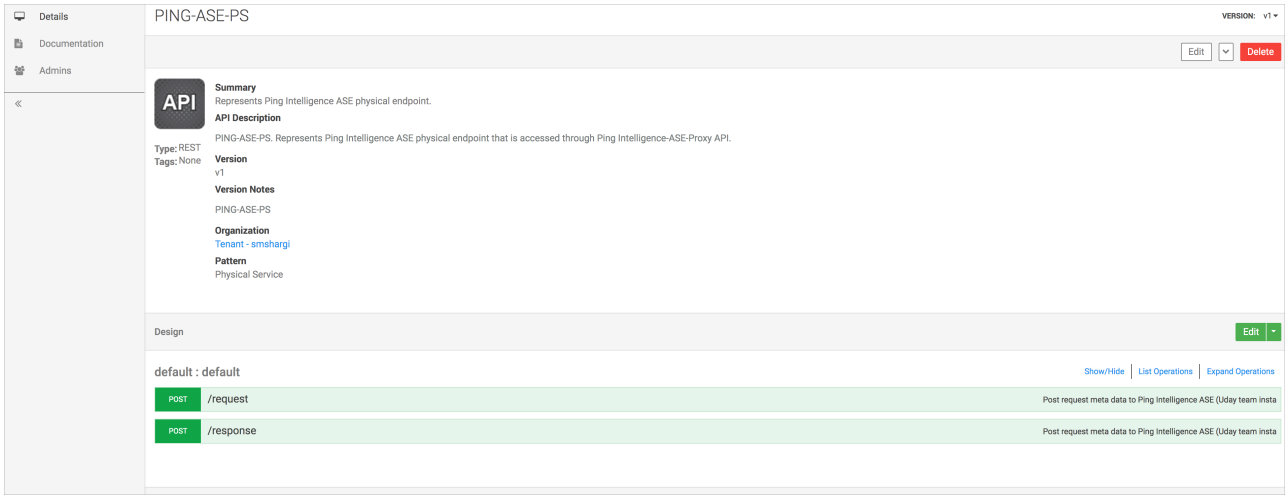
4. Click **Save** after entering the details.



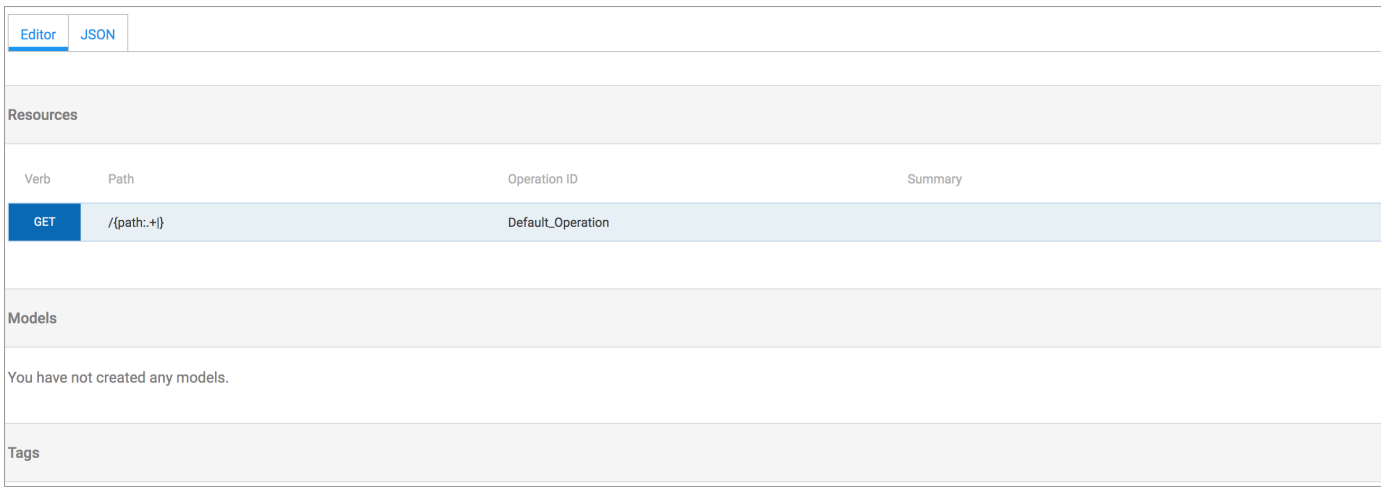
5. Add 2 resources under **Resources**, one to post request metadata to ASE and another to post response metadata to ASE.

   To add a resource to ASE API, open API Designer by performing below 3 steps.

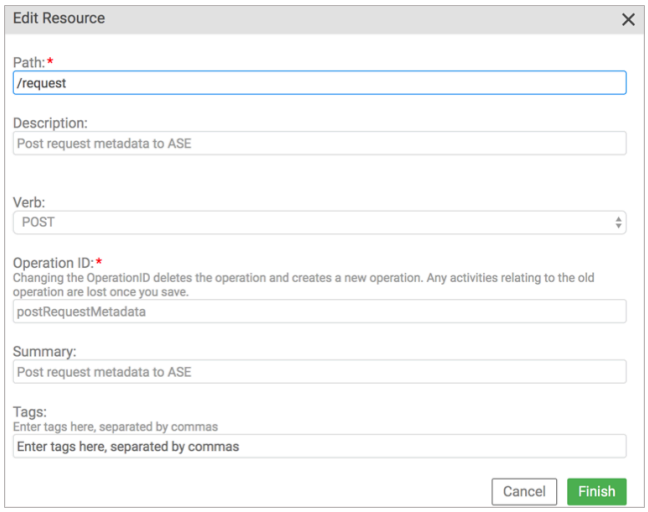
   a. Navigate to the Overview page of the API.
   b. Choose **Details** from the left menu pane. The summary of the API is displayed in the details.
   c. In the **Design** section, click **Edit** to enter API Designer.



   To add the Request resource to API



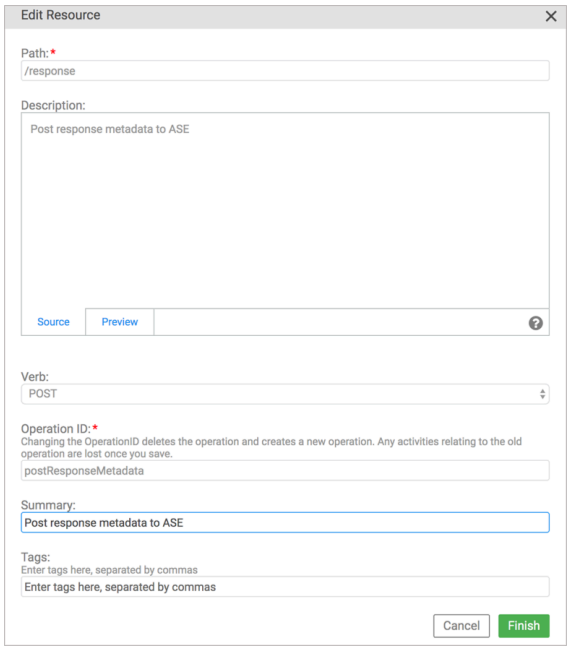
   a. Click **Add Resource** to open the **Edit Resource** window.
   b. Enter `/request` in the **Path** to post request metadata to ASE.
   c. Choose the **Verb**`POST`.
   d. Enter **Operation ID**. If the user does not provide the value, a random value is generated for Operation ID.
   e. Click **Finish** after updating the other non mandatory details like Description, Summary, and Tags.
   f. Click **Save** to reflect the changes.



ⓘ **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.

To add the Response resource to API

a. Click **Add Resource** to open the **Edit Resource** window.
b. Enter `/response` in the **Path** to post request metadata to ASE.
c. Choose the **Verb** `POST`.
d. Enter **Operation ID**. If the user does not provide the value, a random value is generated for Operation ID.
e. Click **Finish** after updating the other non mandatory details like Description, Summary, and Tags.
f. Click **Save** to reflect the changes.



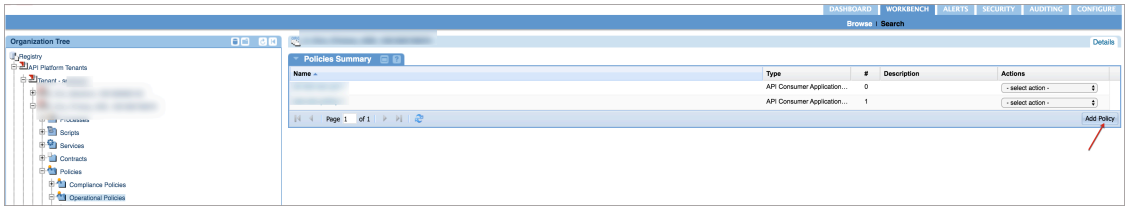
6. Repeat steps 1- 5, and add the secondary (back-up) ASE node.

Secure PingIntelligence ASE APIs

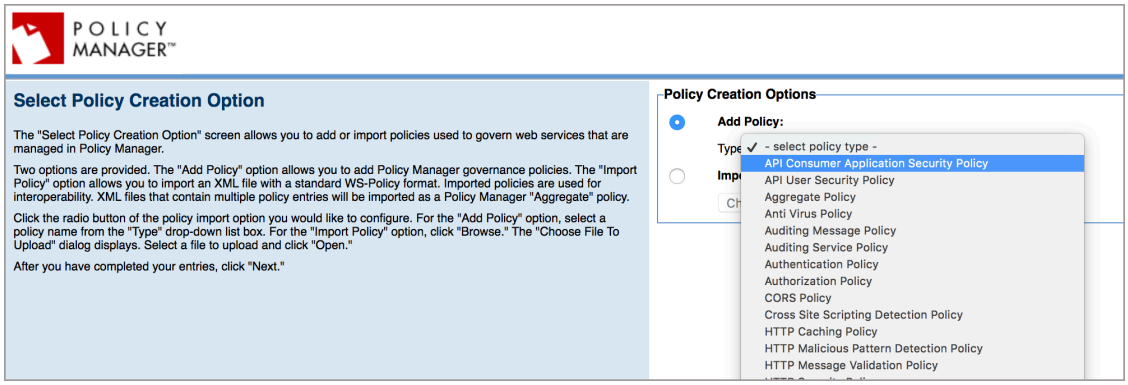
The primary and secondary ASE APIs, added in Akana API Gateway, should be secured from unauthorized access of external clients. To ensure this, secure the ASE APIs using the **API Consumer Application Security operational policy**. The policy allows control on the clients attempting to access the ASE APIs. Add the policy to both primary and secondary ASEs in the Akana gateway.

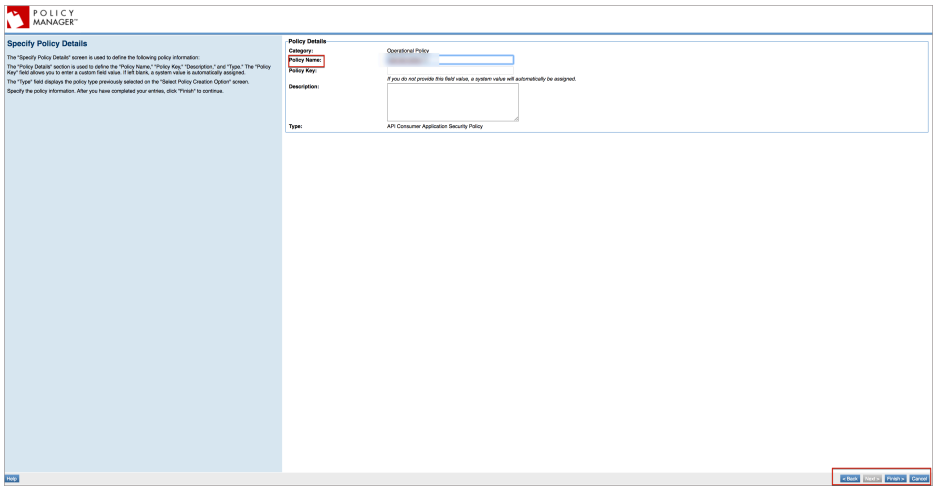
Complete the following steps to add API Consumer Application Security operational policy to ASE APIs::

1. Login 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.

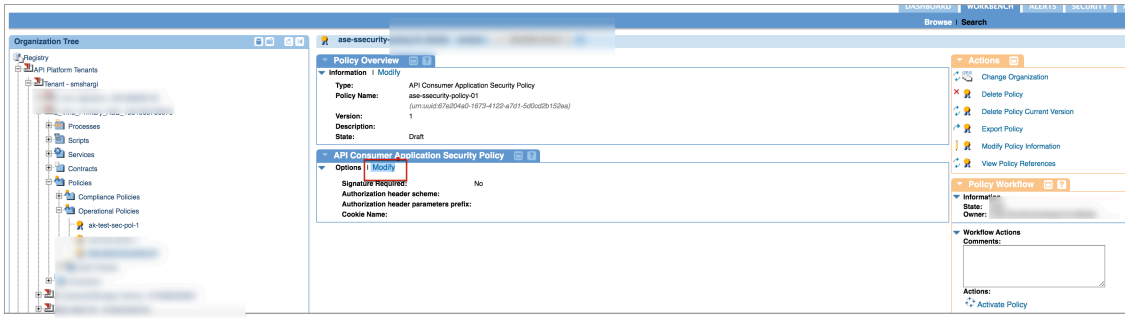


3. In the **Add Policy Wizard**, select **API Consumer Application Security Policy** from **Add Policy**. Enter a **Policy Name**, click **Next** and then click **Finish** to save the policy.

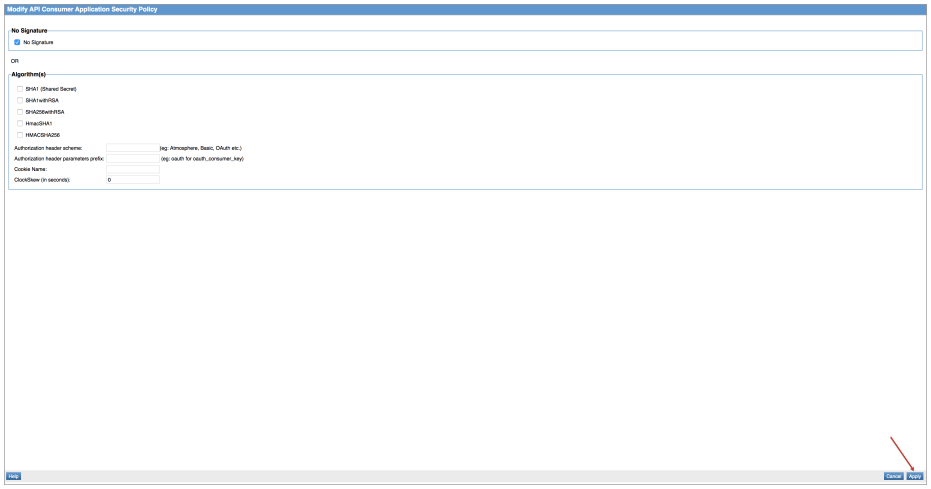




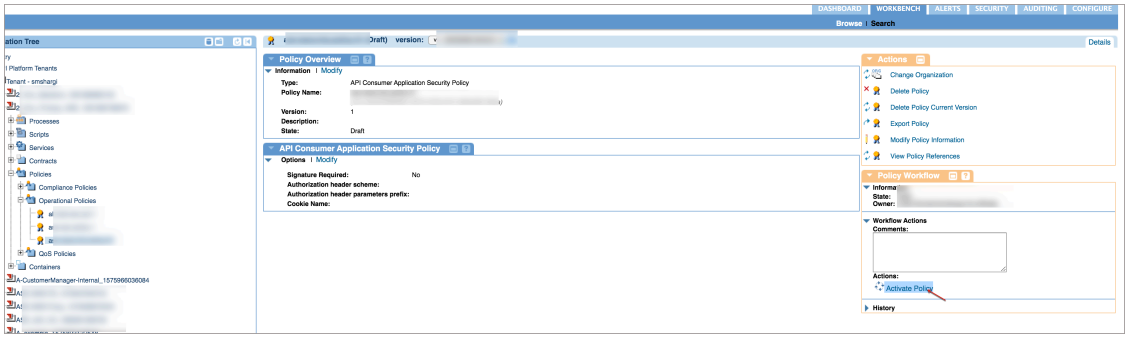
4. The policy appears under **Policies** in ASE API. Click the policy. Click **Modify** in **API Consumer Application Security Policy** section.



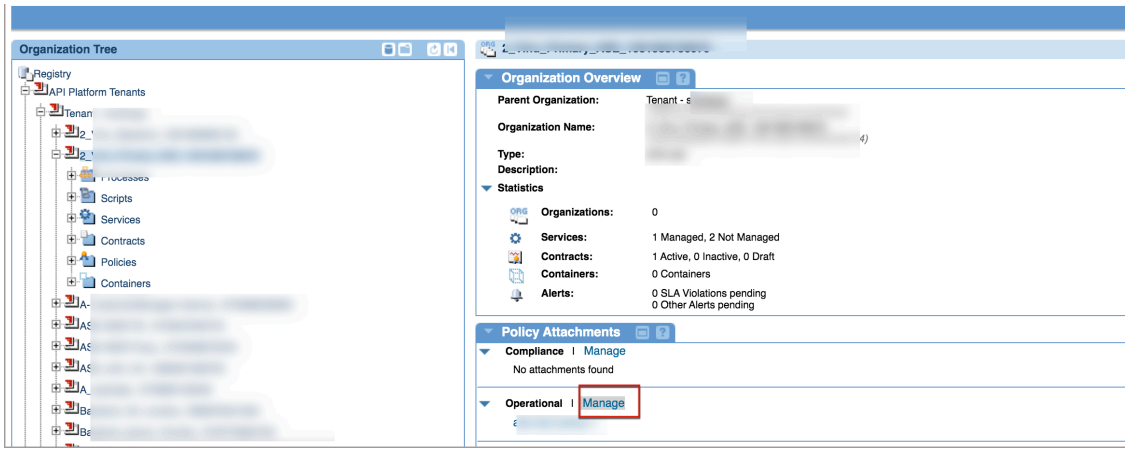
5. Click **Apply** on the next screen without making any changes.



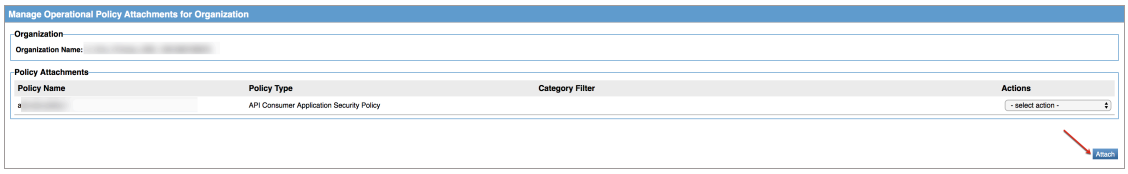
6. Next click **Activate Policy**.



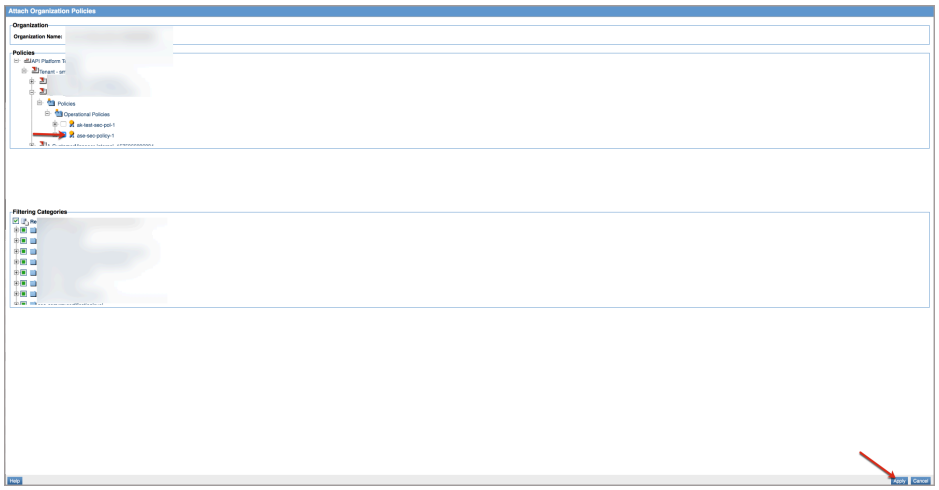
7. Select the ASE API and click **Manage** in **Policy Attachments** section.



8. Click **Attach** on the next page, to attach the policy to ASE API.



9. In the **Attach Organization Policies** window, select the policy added from **Policies** window, and select the **<Policy Name>** checkbox in front of it. Click **Apply**.



10. Click **Close**.
11. Repeat steps 1-10, and add the policy to secondary ASE API.

Capture ASE details

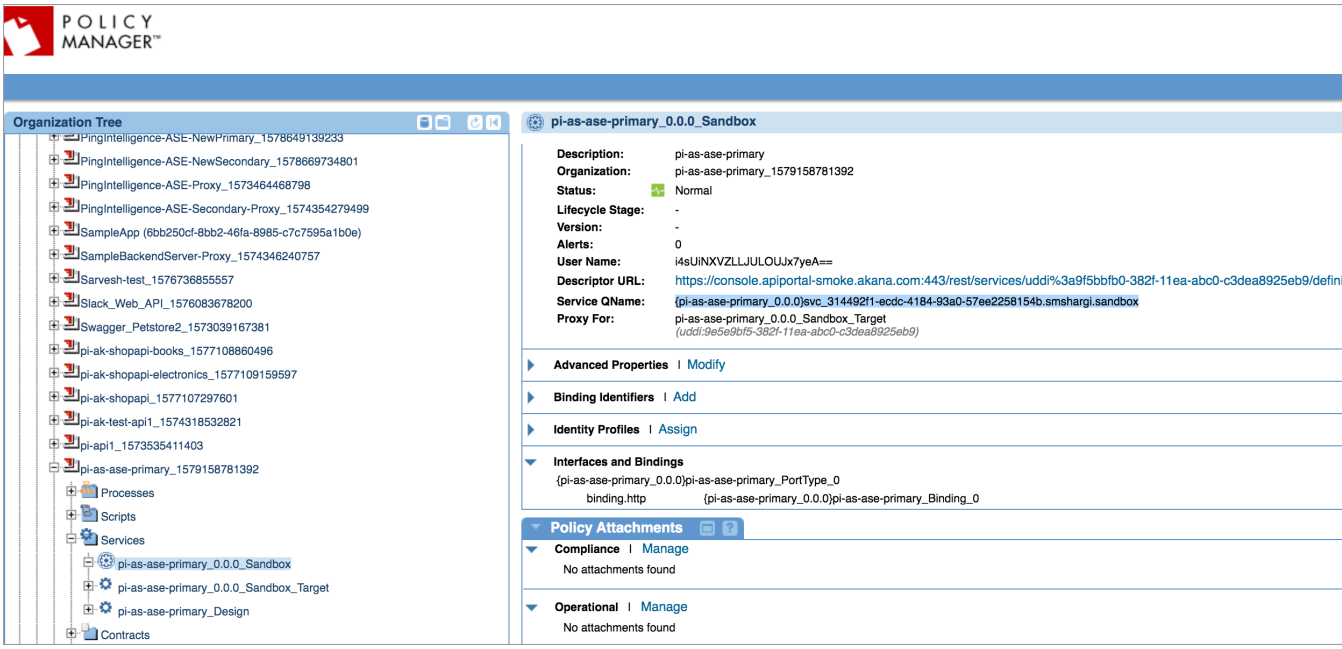
This section elaborates the steps involved to capture details of Service name, Interface name,Operation name for primary and secondary ASE nodes.The Service Qname,Interface Name, and Operation Name are used in **config.js**.

To capture the above values:

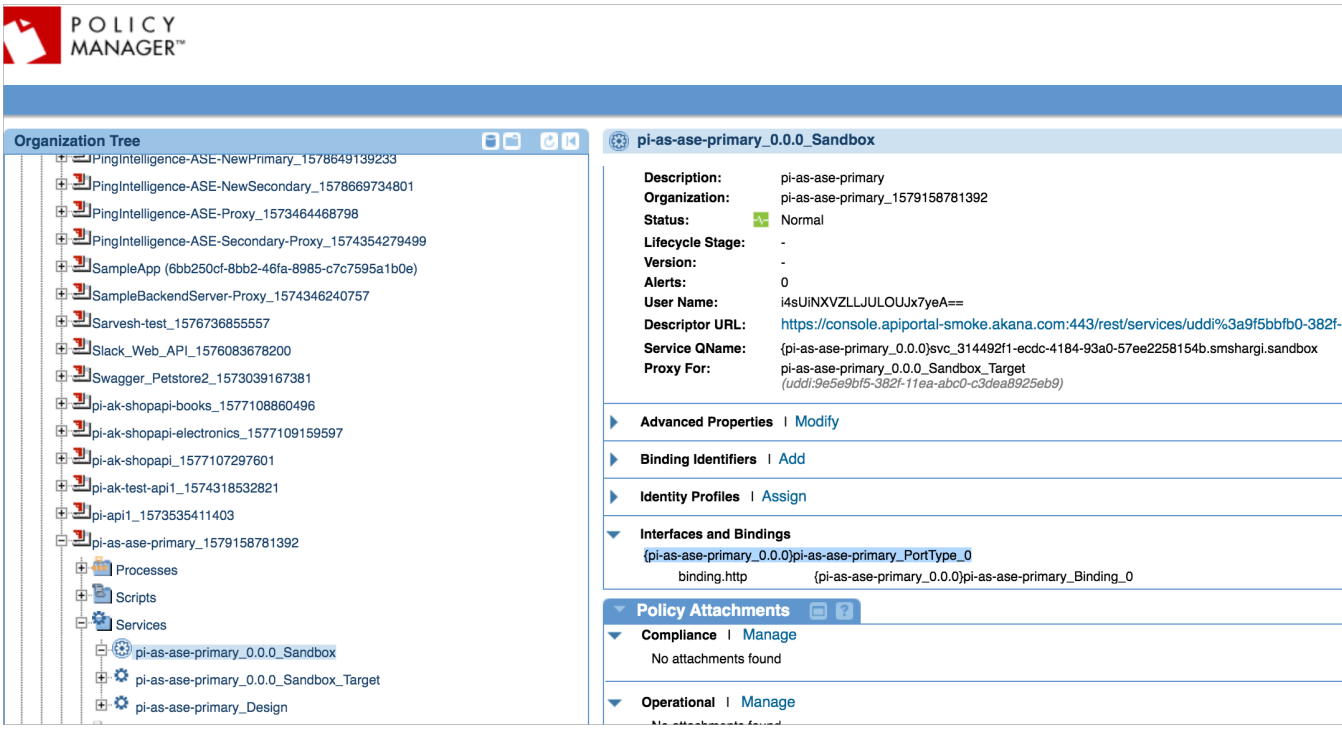
1. Login to **Akana Policy Manager**, navigate through the **Organization Tree** on the left and select the **Tenant** and then the **ASE API**.

2. Expand the **Services** and click on the **API**.

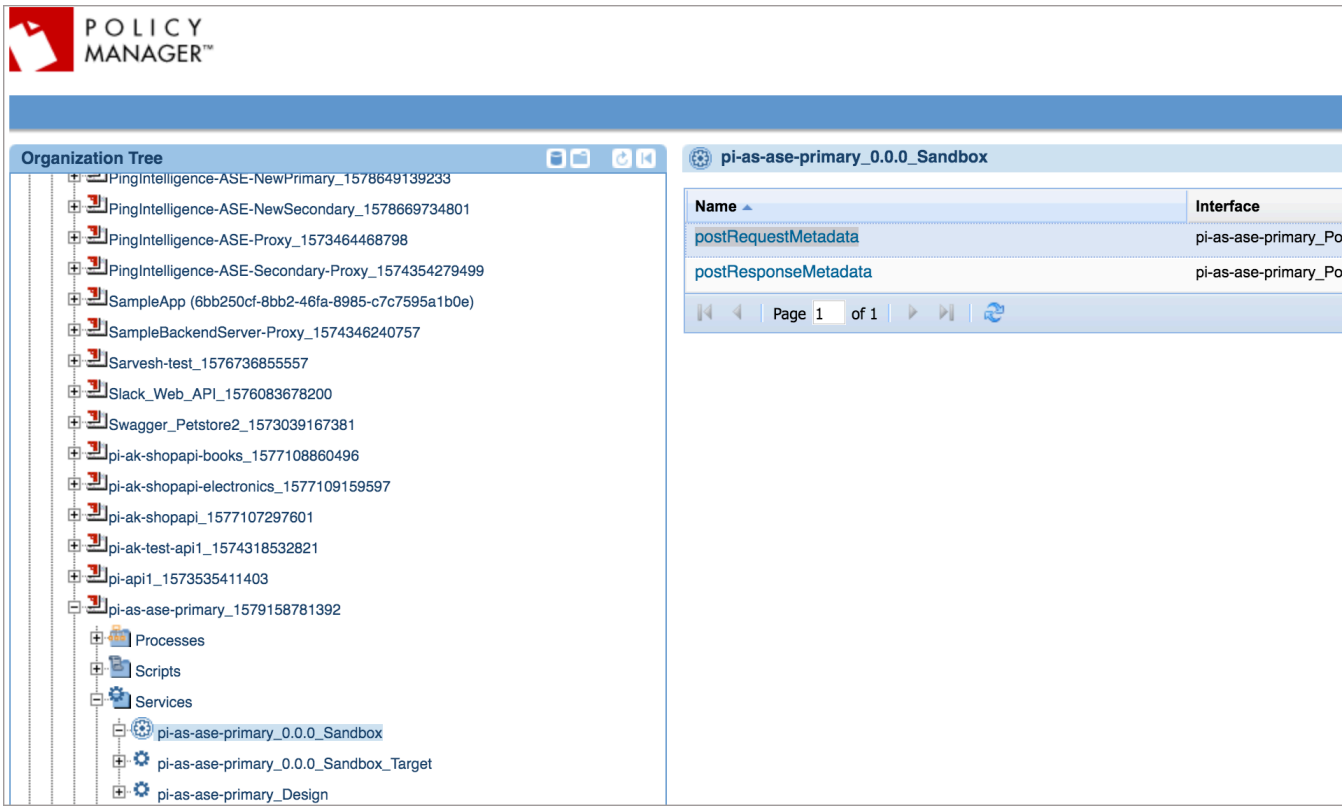
   a. Capture Service Qname.



   b. Capture Interface Name.



   c. Click **Operations** tab on the menu, and capture Operations Name.



ⓘ **Note:** Repeat the above steps to capture the Service Qname,Interface Name, and Operation Name details for Secondary ASE API.

Deploy PingIntelligence policies

Deploying PingIntelligence policies in Akana API gateway is divided into three parts:

- Adding an input script (config.js).
- Adding PingIntelligence policy and applying the policy to APIs.
- Adding RetainerHeader policy and applying the policy to APIs.

Complete the following steps to download and extract the PingIntelligence policy:

1. Download the PingIntelligence policy.
2. Extract the policies by using the following command.

```
# tar -zxvf <<file name>>
```

For example,

```
# tar -zxvf pi-api-akana-policy-4.1.1.tar.gz
```

Add Input script

Complete the following steps to add input script to API gateway:

1. Login to **Akana Policy Manager**, navigate to the **Tenant** and click **Scripts**.
2. Click **Add Script**.

3. Enter **Script Name** and **Script Description**, and click **Next**.

4. Select **JavaScript** as **Language** from the list.
5. Copy the contents of **config.js** script provided by PingIntelligence and paste them into the **Source**.

6. Substitute the values of ‘Service_QName’, ‘Interface_Name’, and ‘Operation_Name’ that were captured in Capture ASE details step.This needs to be performed for both primary and secondary ASE nodes. The below table lists the variables in **config.js** that needs to be populated.

Variable	Purpose
ase_token	Variable to hold 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.

Here is a sample substitution snippet for reference:

```
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";
```

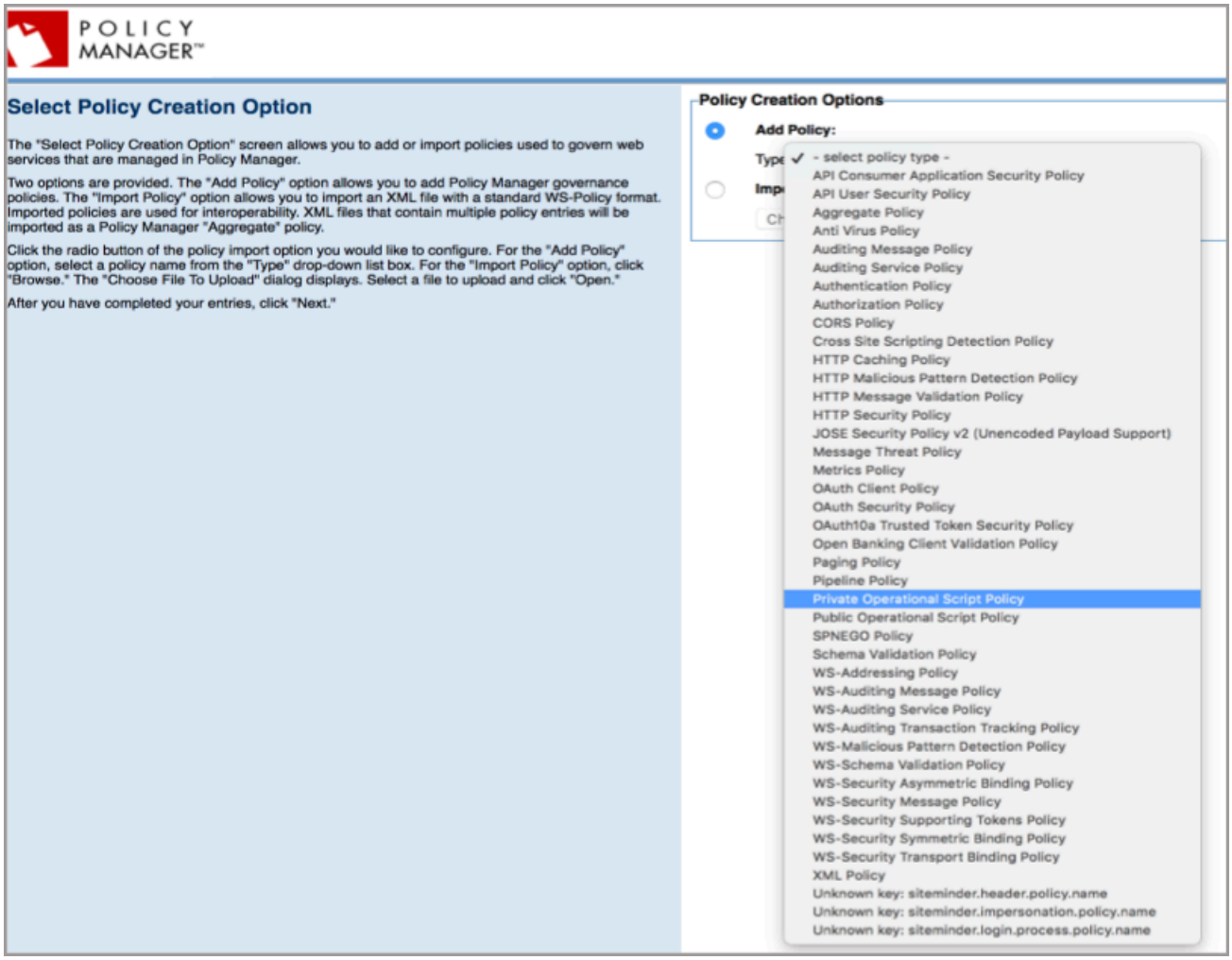
7. Click **Finish**, and then click **Close**.

Add PingIntelligence policy

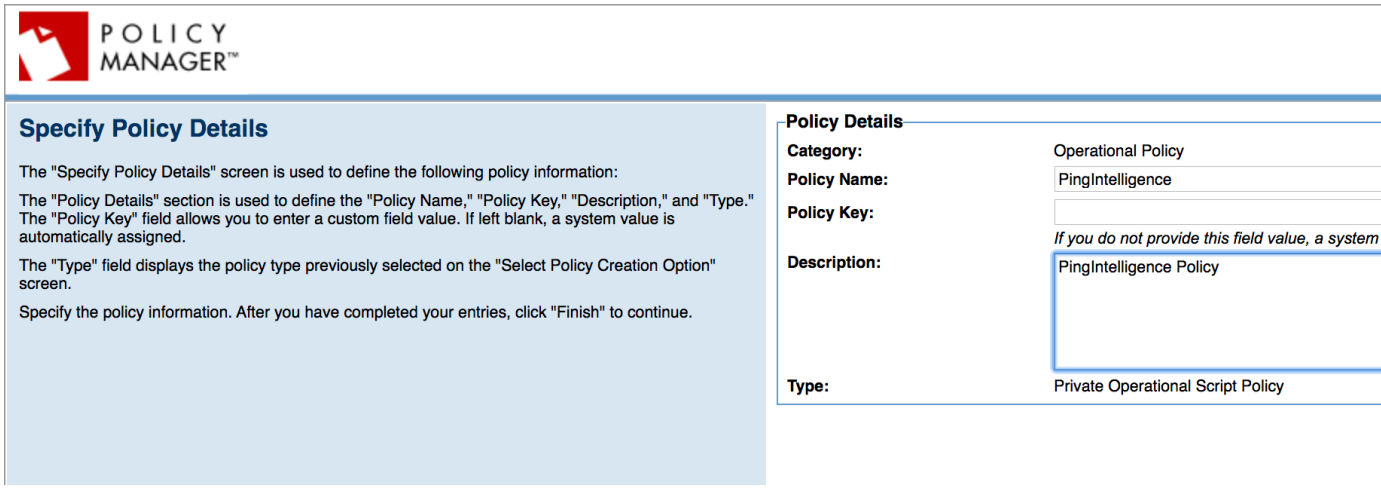
Complete the following steps to add a PingIntelligence policy to Akana gateway:

1. Login to **Akana Policy Manager** and navigate to the **Tenant**. Under **Policies** click **Operational Policies**.

2. Select **Add Policy** option. Select Policy Type as **Private Operational Script Policy** from the list and click **Next**.

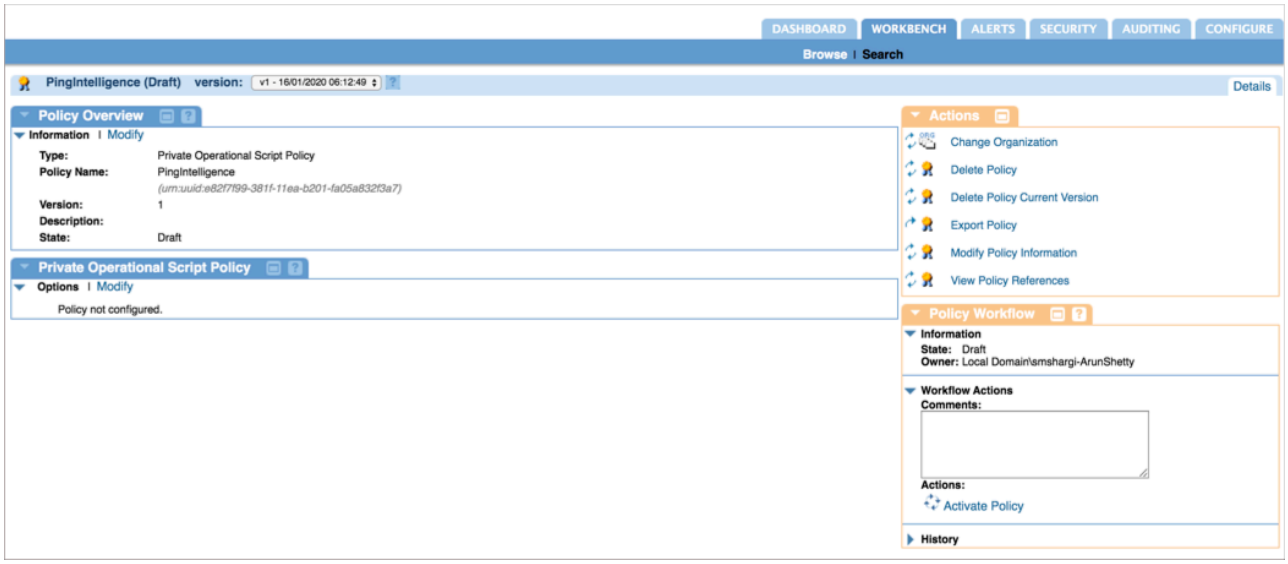


3. Provide **Policy Name** and **Description**, click **Finish** and then click **Close**.

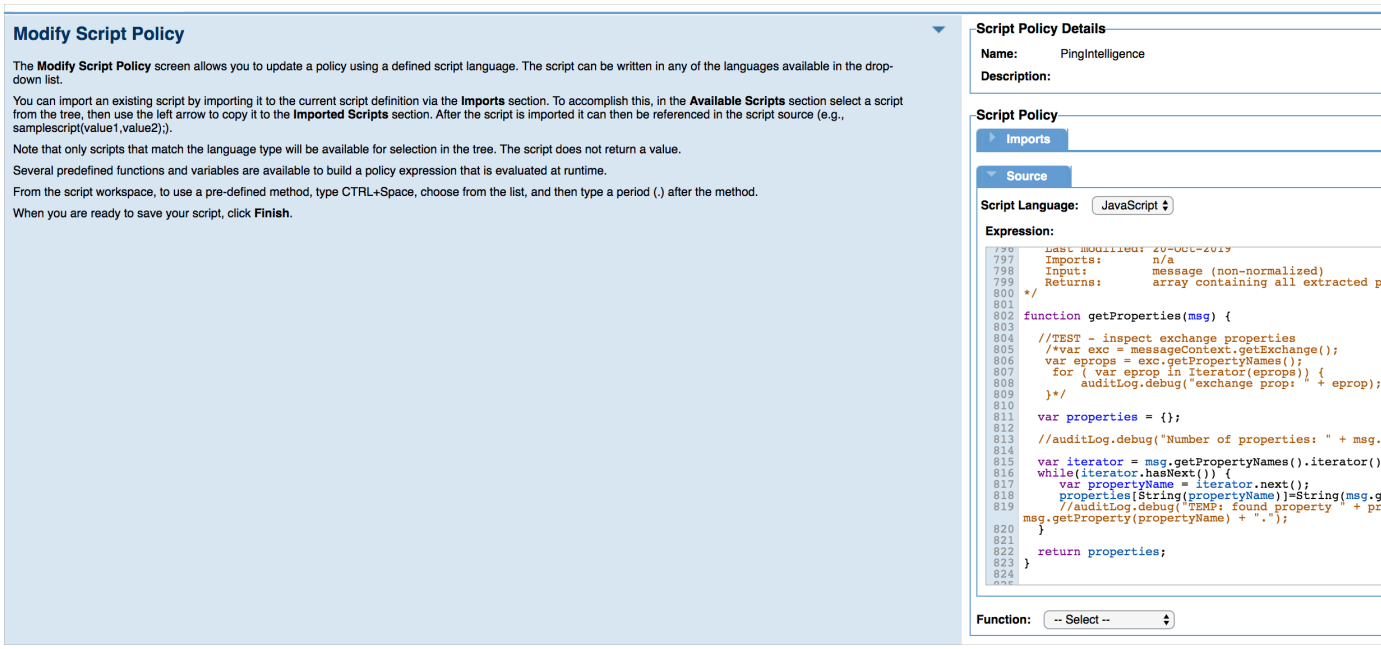


4. Navigate to **Workbench**.

5. In the **Private Operational Script Policy** section, select the policy name and click **Modify**.



6. Click on **Imports**. Select the script added in Add Input Script step above and import it by clicking **<<**.
7. Select **JavaScript** as language **Language** from the list.
8. Copy the contents of **pi_policy.js** script and paste them into **Expression** in **Source**.



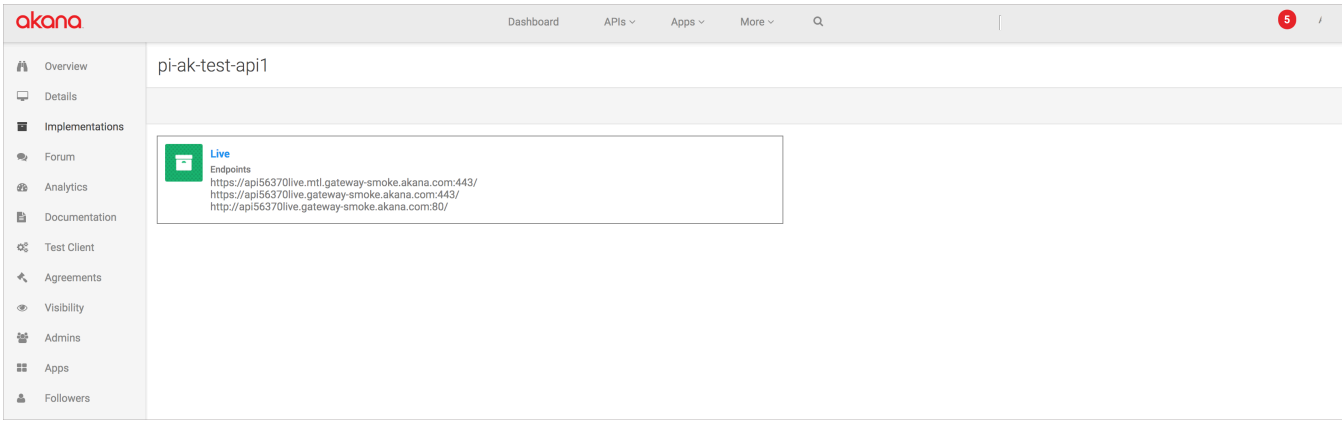
9. Click **Finish** and then click **Close**.
10. In the **WorkFlow Actions** click **Activate Policy** to activate the PingIntelligence policy.

Apply the PingIntelligence policy to APIs

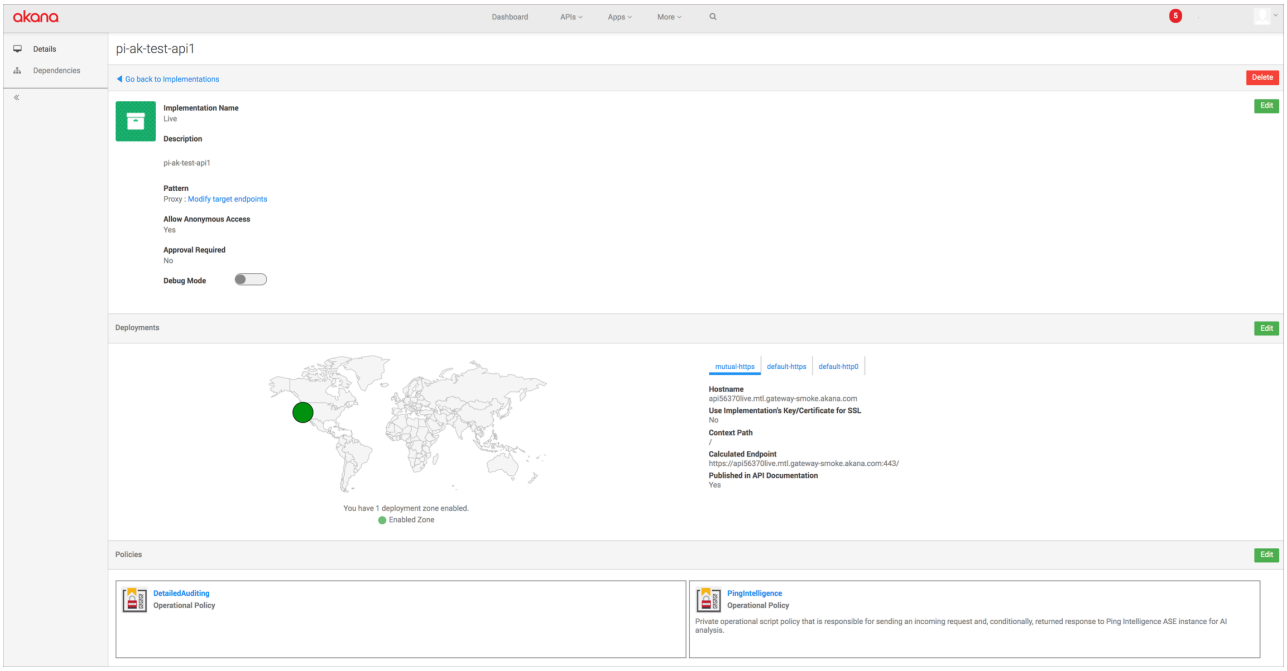
The PingIntelligence Policy can be applied at tenant level, org level and at individual API level. The following steps explain the process of adding a policy at API level.

1. Login to **Akana Portal**.

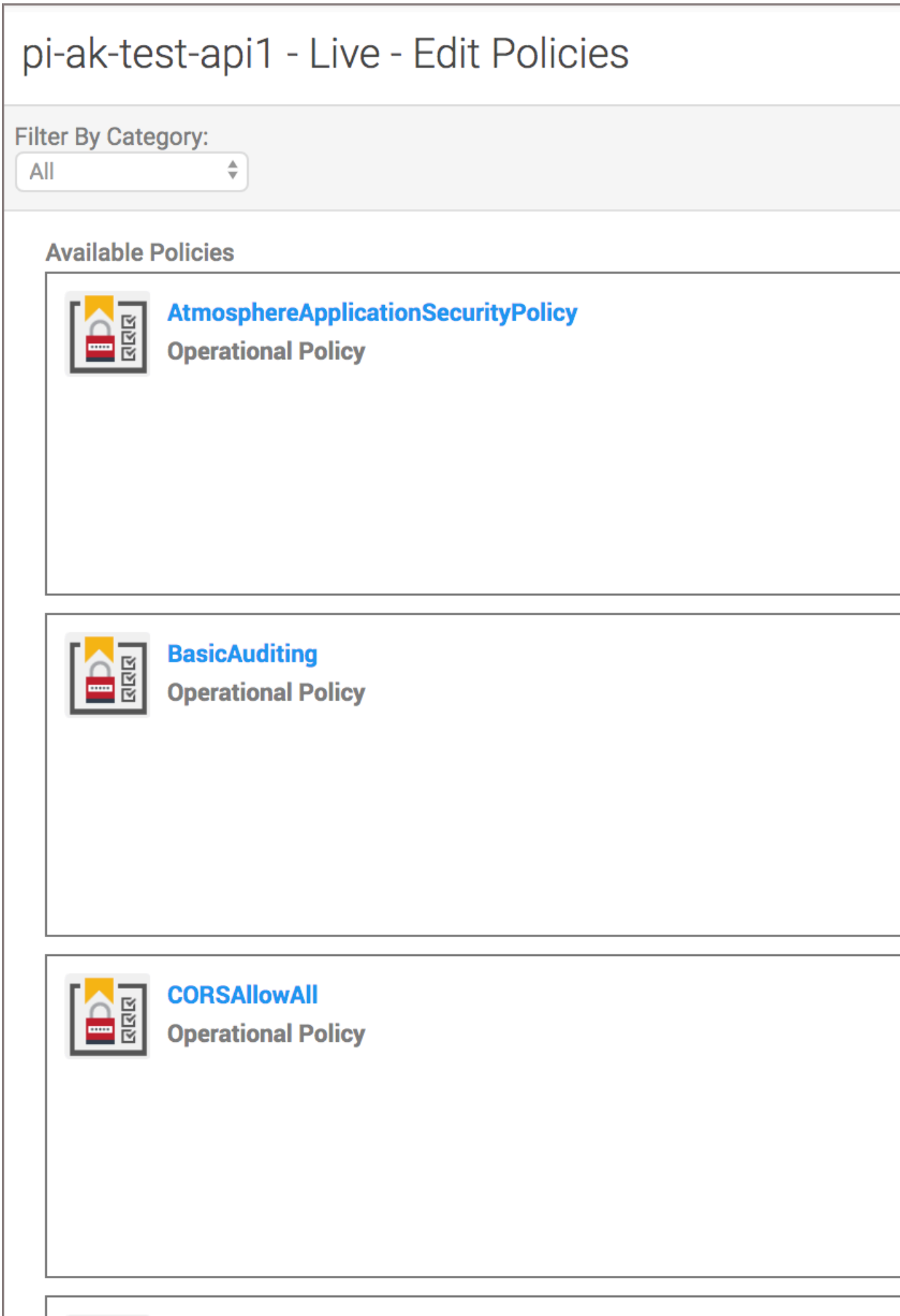
2. To apply policy at per API level:
   a. Click on the API name.
   b. In the left window pane, click **Implementation** on the left pane.



   c. Click **API implementation name** icon. Possible values for API implementation could be (Live/ Sandbox/Development).
   d. Click **Edit** in **Policies** section.



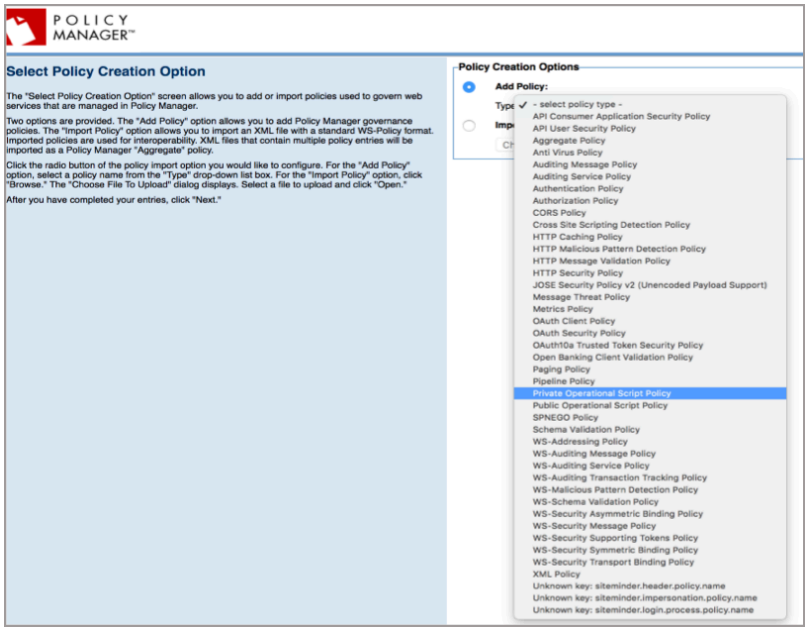
   e. Find the PingIntelligence Policy in **Available Policies** pane Click **Attach** under **PingIntelligence** policy.
   f. Click **Save**.



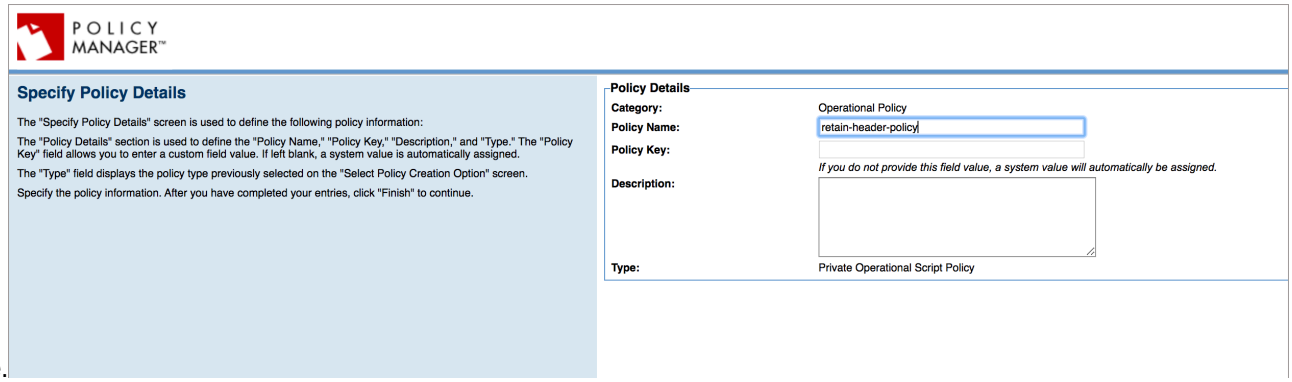
Add RetainHeader policy

Complete the following steps to add RetainHeader Policy to Akana Gateway:

1. Login to **Akana Policy Manager** and navigate to the **Tenant**. Under **Policies** click **Operational Policies**.
2. Select **Add Policy** option. Select Policy Type as **Private Operational Script Policy** from the list and click **Next**.



3. Provide **Policy Name** and **Description**, click **Finish** and then click **Close**.



4. Navigate to **Workbench**.

5. In the **Private Operational Script Policy** section, select the policy name and click **Modify**.

RetainHeaderValue (Draft) **version:** v2 - 10/02/2020 07:30:29

Policy Overview

Information | Modify

Type:	Private Operational Script Policy
Policy Name:	RetainHeaderValue
	(urn:uuid:b7d5e23b-0d00-11ea-a91b-9d6dbab54157)
Version:	2
Description:	Retains value for Authorization header and copies to X_Au
State:	Draft

Private Operational Script Policy

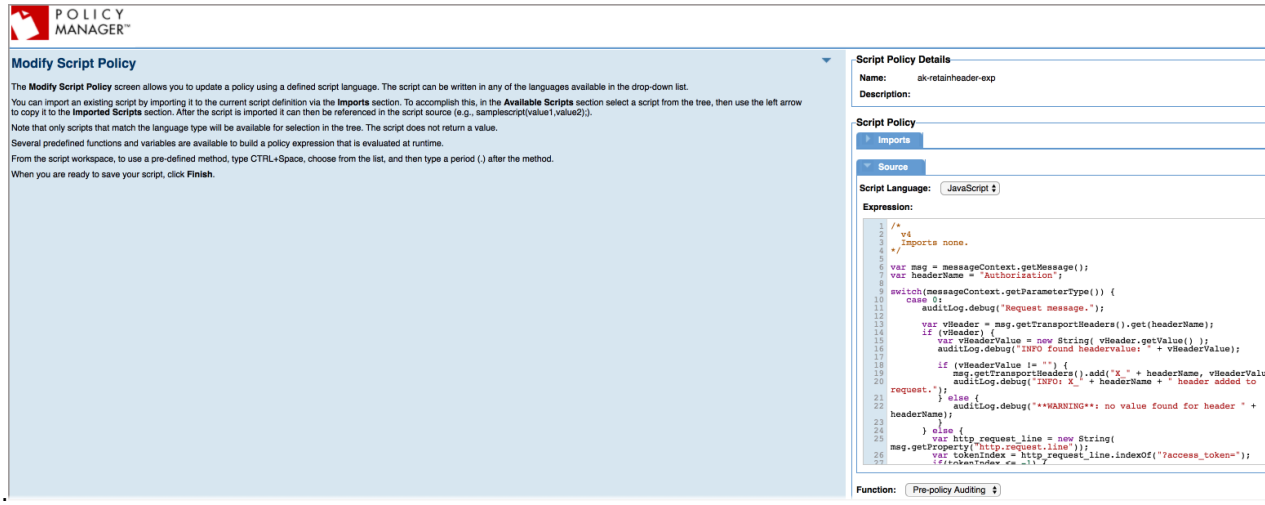
Options | Modify

Function:	Pre-policy Auditing
Script Language:	JavaScript

Expression:

```
/* v4 Imports none. */ var msg = messageContext.getMessage(); var headerNam
vHeader = msg.getTransportHeaders().get(headerName); if (vHeader) { var vHe
(vHeaderValue != "") { msg.getTransportHeaders().add("X_" + headerName, vHe
auditLog.debug("**WARNING**: no value found for header " + headerName); } }
```

6. Select **JavaScript** as language in **ScriptLanguage** from the list.
7. Copy the contents of **retain-header-policy.js** script and paste them into **Expression** .
8. Select **Pre-policy auditing** from **Function**.



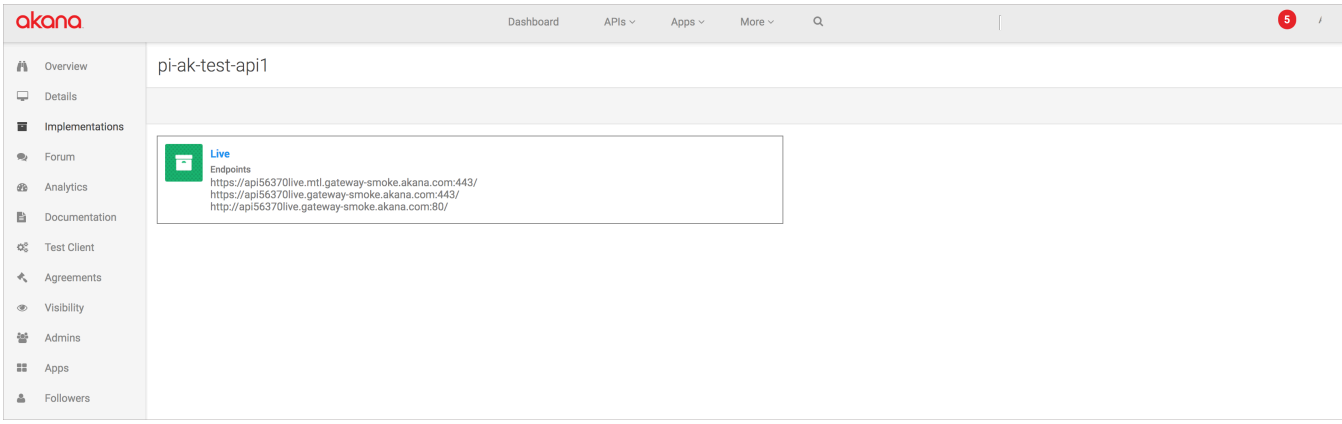
9. Click **Finish** and then click **Close**.
10. In the **WorkFlow Actions** click **Activate Policy** to activate the RetainHeader policy.

Apply the RetainHeader policy to APIs

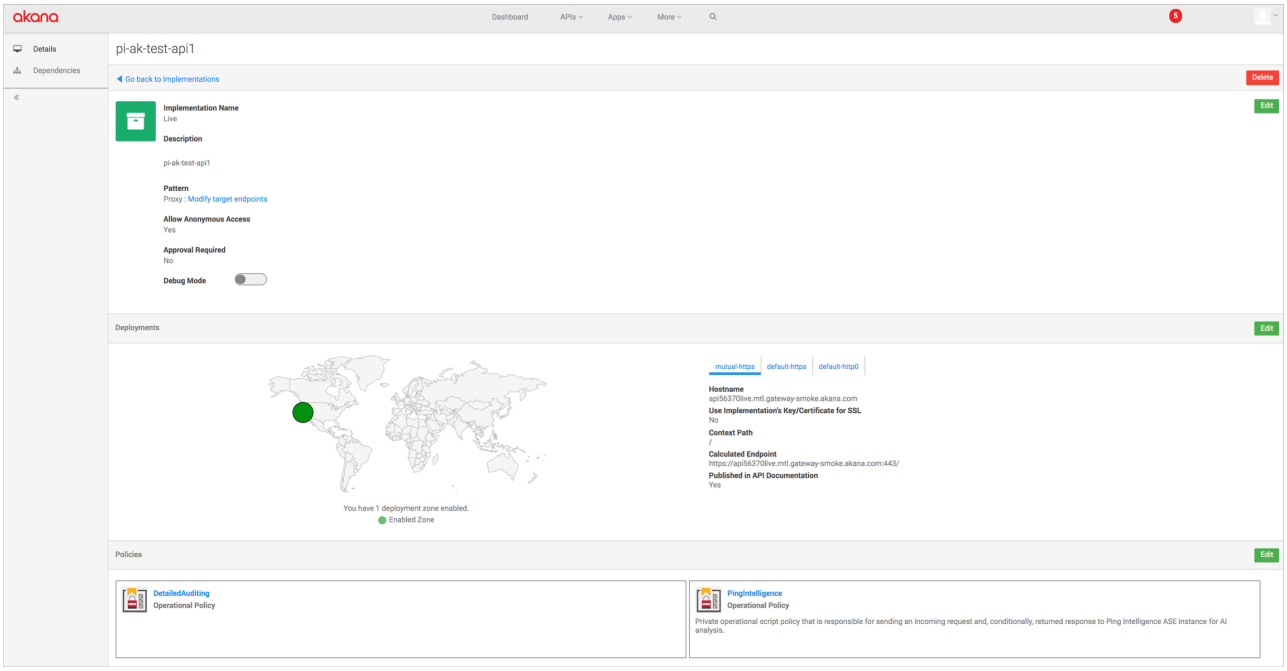
The RetainHeader Policy can be applied at tenant level, org level and at individual API level. The following steps explain the process of adding a policy at API level.

1. Login to **Akana Portal**.

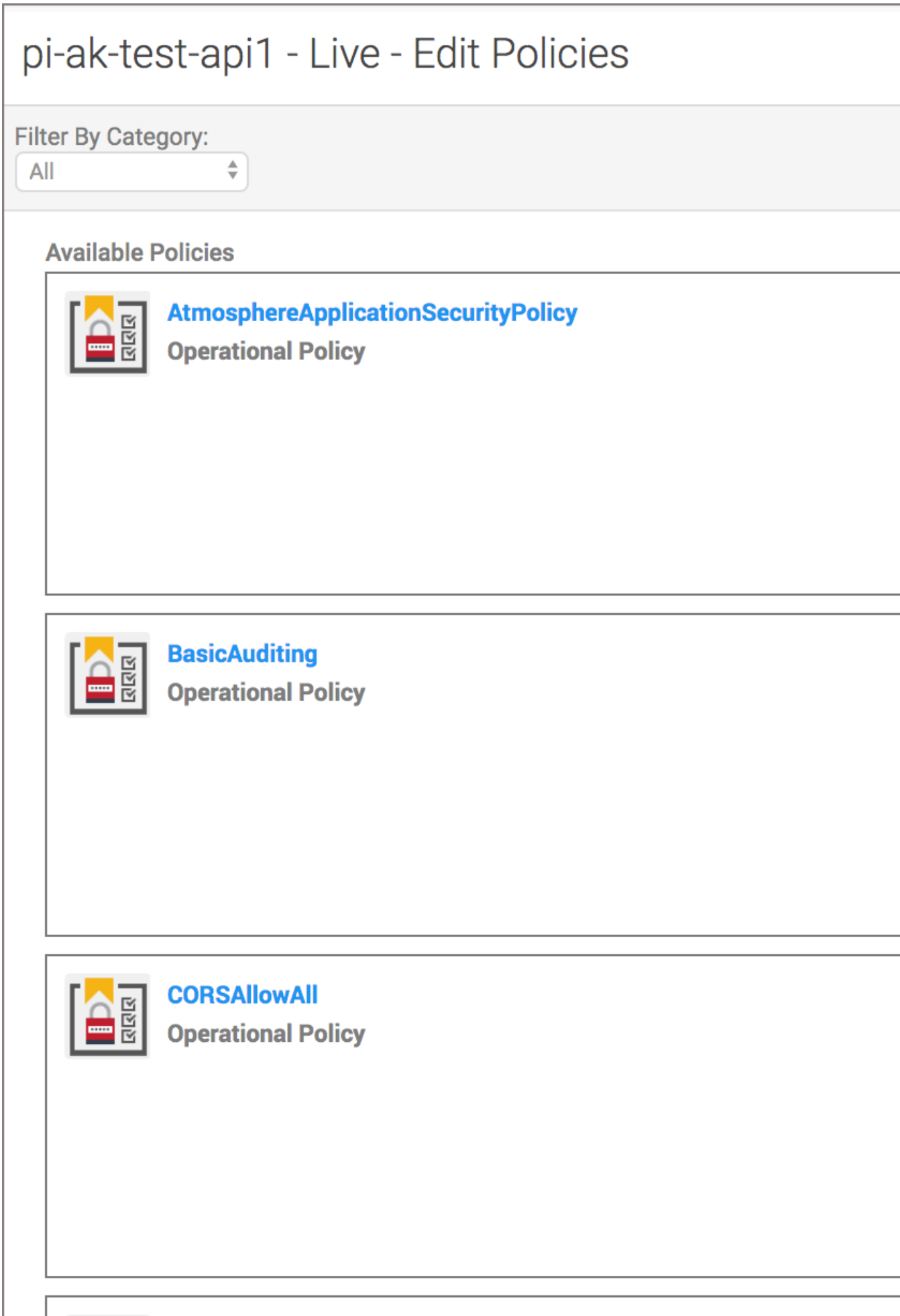
2. To apply policy at per API level:
   a. Click on the API name.
   b. In the left window pane, click **Implementation** on the left pane.



   c. Click **API implementation name** icon. Possible values for API implementation could be (Live/ Sandbox/Development).
   d. Click **Edit** in **Policies** section.



   e. Find the RetainHeader Policy in **Available Policies** pane Click **Attach** under **RetainHeader** Policy.
   f. Click **Save**.



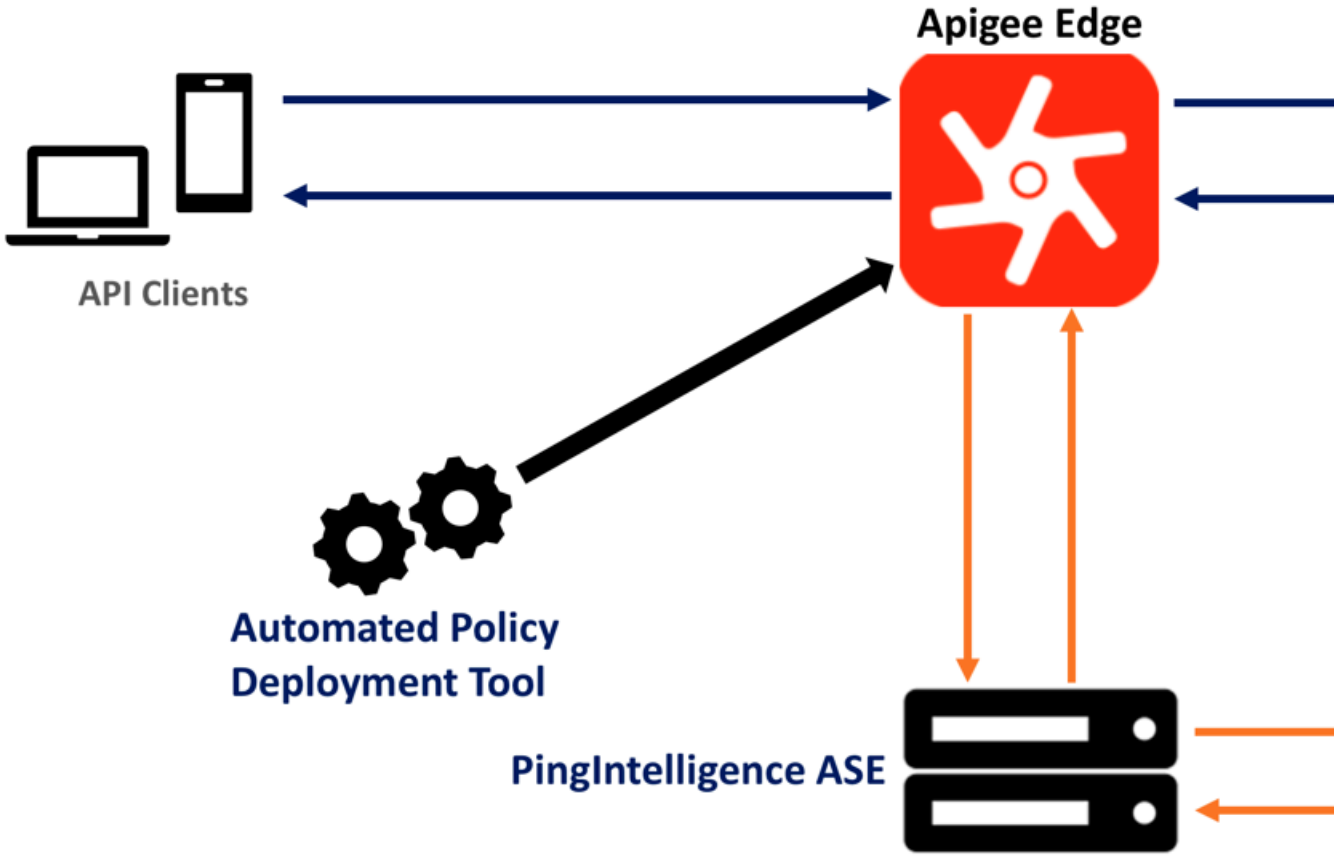
Apigee API gateway integration

PingIntelligence 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 callout and flow hook policies. A Flow Hook in Apigee Edge applies the PingIntelligence shared flow globally to all APIs in an environment under an organization. The Flow Call Out policy in Apigee Edge applies the PingIntelligence shared flow on a per API basis in an environment under an organization.

PingIntelligence provides an automated tool to deploy both Flow Hook and Flow Call Out polices.

The following diagram shows the logical setup of PingIntelligence ASE and Apigee



Edge:

Here is the traffic flow through the Apigee Edge and PingIntelligence for APIs components.

1. Incoming request to Apigee Edge
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 IP, cookie, OAuth2 token or API key against the Blacklist. 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 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.
5. The response from the backend server is received by Apigee Edge.
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.

Prerequisites to deploying PingIntelligence shared flow

Confirm that the following prerequisites are met before using the PingIntelligence Apigee tool.

Prerequisite:

- **Apigee version** - PingIntelligence 4.0 and later works with Apigee Edge Cloud 18.12.04
- The machine where the PingIntelligence Apigee tool is installed has anyone of OpenJDK versions between 11.0.2 to 11.0.6 installed.
- **PingIntelligence software installation**

  PingIntelligence 4.0 or later software are installed and configured. For installation of PingIntelligence software, see the manual or platform specific automated deployment guides.
- **Verify that ASE is in sideband mode**

  Make sure that in ASE is in `sideband` mode by running the following command in the 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                      : 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.
- **Enable sideband authentication:** 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
  ```
- **Generate sideband authentication token**

  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 PingIntelligence automated policy tool. To generate the token in ASE, enter the following command in the ASE command line:

  ```
  # ./bin/cli.sh -u admin -p admin create_sideband_token
  ```

  Save the generated authentication token for further use.

Download automated policy tool

Download

Complete the following steps to download and install the PingIntelligence policy tool:

1. *Download* the PingIntelligence policy tool to the `/opt` directory.

2. Complete the following steps to untar the policy tool:
   a. At the command prompt, type the following command to untar the policy tool file:

      ```
      tar –zxvf <filename>
      ```

      For example:

      ```
      tar –zxvf pi-apigee-4.1.tar.gz
      ```

   b. To verify that the tool successfully installed, type 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 and configured Apigee credentials.
`lib`	Jar files and various dependencies. Do not edit the contents of this directory.
`policy`	Contains the shared flows: ▪ `request_sharedflow.zip`: Shared flow policy for request ▪ `response_sharedflow.zip`: Shared flow for response. Contains the self-signed certificate that is shipped by default with ASE. The name of the file is `ase32.pem`.
`config`	Contains the `apigee.properties` file.
`logs`	Contains the log and status files.

To configure the PingIntelligence policy tool after installation, edit the `apigee.properties` file and set the necessary properties. For more information, see *Configure Apigee properties file* on page 515.

Configure Apigee properties file

The `apigee.properties` file is required for all sideband Apigee configurations. It is used to set properties for PingIntelligence policy tool after installation. It can also be optionally configured to capture the user information. The file is available in the `/pingidentity/pi/apigee/config/` directory.

The following table describes the variables in the file.

Variable	Description
`configuration_store`	Choose where ASE token is stored. The possible values are `kvm` and `custom`. The default is `custom`. When `custom` is chosen, 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.

Variable	Description
`apigee_url`	URL to connect to Apigee Edge ⓘ **Note:** If your Apigee installation is on a private cloud, then change the URL to the one that matches your Apigee management server API IP:Port or hostname with protocol.
`apigee_username`	Username to connect to Apigee Edge
`apigee_password`	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 during the *prerequisite* step.
Configuration properties to extract user information	
`enable_oauth_policy`	Choose whether to send user_info to ASE or not using the PingIntelligence OAuth Policy. Possible values are `true` or `false`. The default value is `false`. When set to `false` the OAuthPolicy is not executed. For more information on OAuthPolicy, see *Extract user information from access tokens* on page 519.
`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. For example. `access_token_position=queryparam` ⓘ **Note:** At present only **Bearer** prefix is supported in Authorization header by Apigee.

Variable	Description
`access_token_variable`	A variable to hold `access_token` value. The default value is `Authorization`. It is used in the OAuthPolicy. For example, `access_token_variable=access_token => -H "access_token: Rft3dqrs56Blirls56a"`
`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 `access_token` info. The default value is `client_id`.
Timeout configurations	
`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.
`keepalive_timeout`	Connection keepalive timeout between Apigee API gateway and PingIntelligence ASE. Make sure that enable_keepalive to `true` in `ase.conf` for keepalive configuration to take effect. ⓘ **Note:** Make sure that the enable_sideband_keepalive is set to `true` in `ase.conf` file for keepalive connection between Apigee API gateway and ASE. For more information, see *ASE configuration - ase.conf* on page 146.

ⓘ **Note:** Backslashes ' \' are not supported in `username` and `client_id` values.

The following is a sample `apigee.properties` file.

```
# 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)
# Setting it to false will not send user_info to ASE
enable_oauth_policy=false
# 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
# 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 above configuration will be embedded into the PingIntelligence policy. If `configuration_store` is set to `kvm`, the above configuration is pushed to a key-value map store while deploying the policy and is retrieved during policy execution.

Reset timeout configurations

About this task

You can reset the timeout configurations after you have deployed the PingIntelligence policy in the following two ways:

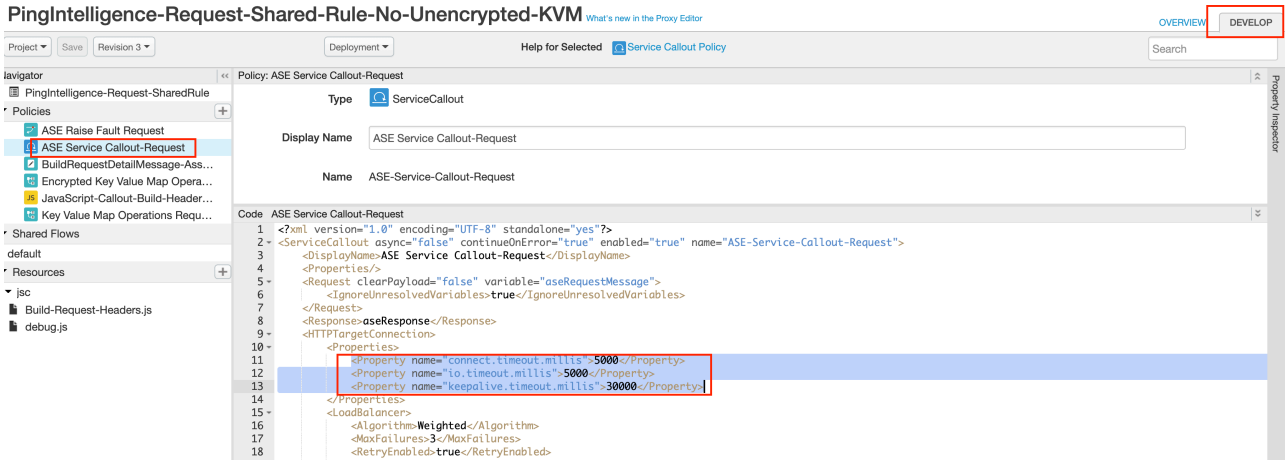
- Undeploy the policy and reset the values in `apigee.properties` file and redeploy the PingIntelligence policy. For more information on undeploying the policy, see *Change deployed policy mode* on page 529. Or

- Update the values in Apigee Edge Management UI.

Following are the steps to update the timeout configurations in Apigee Edge Management UI:

Steps

1. In the **Apigee Edge Management** UI, navigate to **Sharedflows** page.
2. In the **Sharedflows** page, open **PingIntelligence-Request-SharedRule**.
3. Click on **ASE Service Callout-Request** under **Policies**.
4. Click on 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** also.

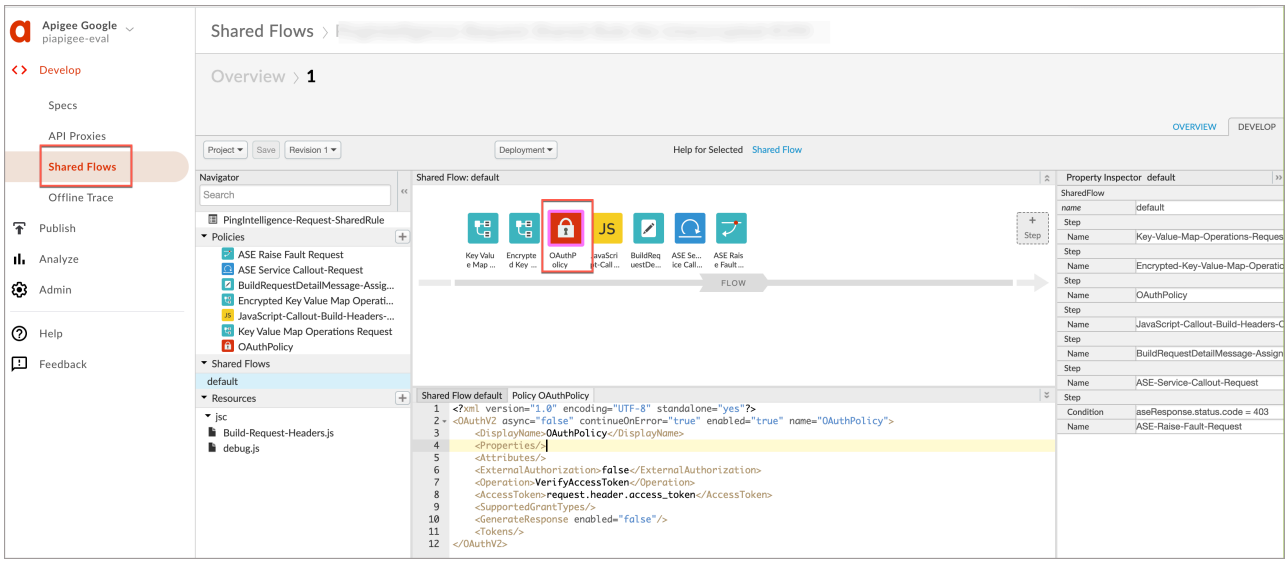
Extract user information from access tokens

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 Call Out. For more information, see *Deploy PingIntelligence Policy for Flow Hook* on page 520 and *Deploy PingIntelligence Policy for Flow Call Out* on page 522.

It is advised to deploy the OAuthPolicy.xml using a Flow Call Out policy to leverage the flexibility of applying on a Per API basis. For more information, see *Configure PingIntelligence Flow Call Out in Apigee* on page 523. The following screenshot 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 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`.

Deploy the PingIntelligence policy

Using the PingIntelligence automated policy tool, you deploy the shared flow either by Flow Hook or the Flow Call Out policy which is configured in the command line. Choose either the included ASE self-signed certificate or a CA signed certificate

Deploy PingIntelligence Policy for Flow Hook

With a Flow Hook, the PingIntelligence shared flow is applied to all APIs in the environment of an organization.

Deploy with self-signed certificate: Run the following command to deploy the PingIntelligence policy with self-signed certificate:

```
/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
```

Deploy with CA signed certificate: Run the following command to deploy the PingIntelligence policy with CA-signed certificate:

```
/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
```

Verify the status

After deploying the Flow Hook using the PingIntelligence tool, check the status of the deployment by entering 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
```

Deploy PingIntelligence Policy for Flow Call Out

In the Flow Call Out, the PingIntelligence policy is applied on an per API basis in the environment of an organization.

Deploy with self-signed certificate: Run the following command to deploy the PingIntelligence policy with self-signed certificate:

```
/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
```

Deploy with CA signed certificate: Run the following command to deploy the PingIntelligence policy with CA-signed certificate:

```
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
```

Verify the status

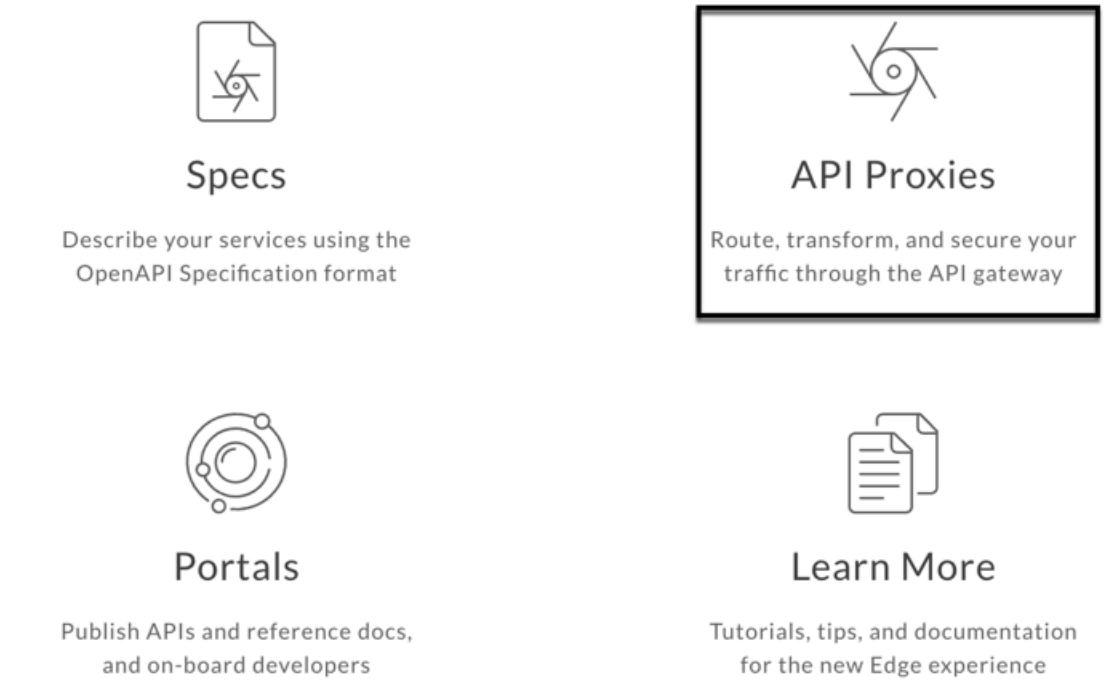
After deploying the Flow Call Out using the PingIntelligence tool, check the status of the deployment by entering the following command:

```
/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
```

Configure PingIntelligence Flow Call Out in Apigee

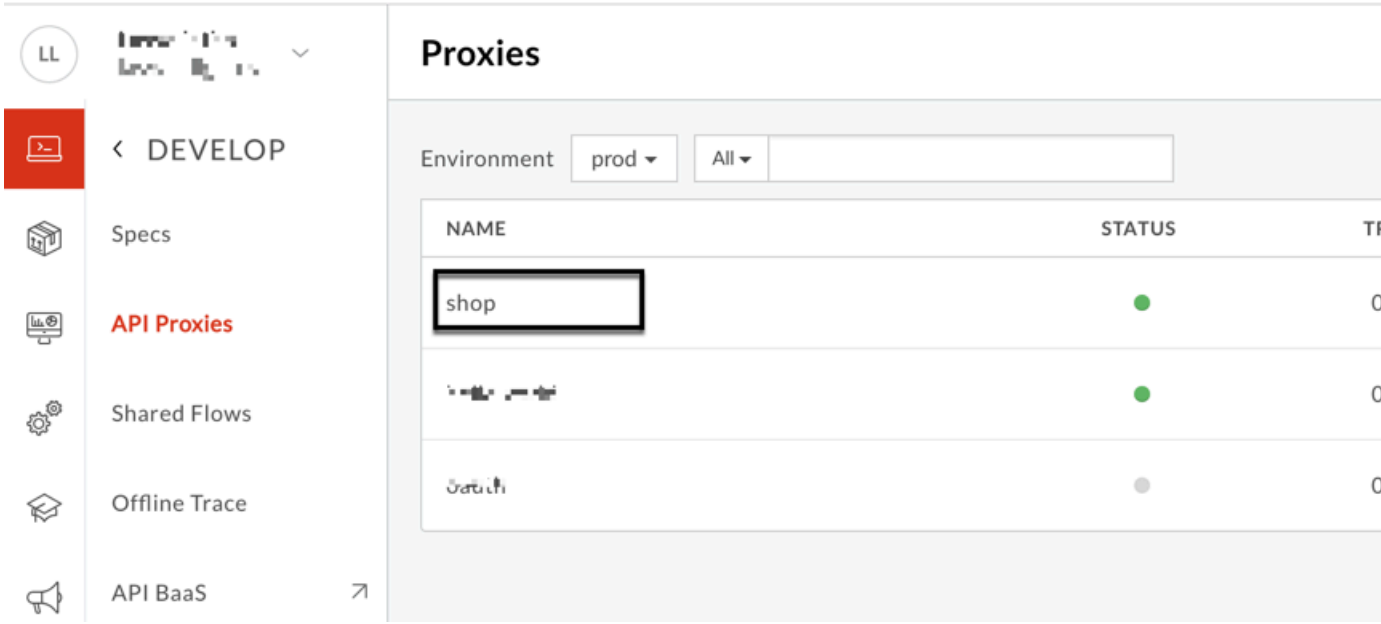
After deploying the Flow Call Out policy using PingIntelligence, configure the PingIntelligence for APIs shared flow. Complete the following steps for Flow Call Out for request and response. The steps listed are for request, complete the same steps for response.

1. Log in to your Apigee Edge account and choose the API

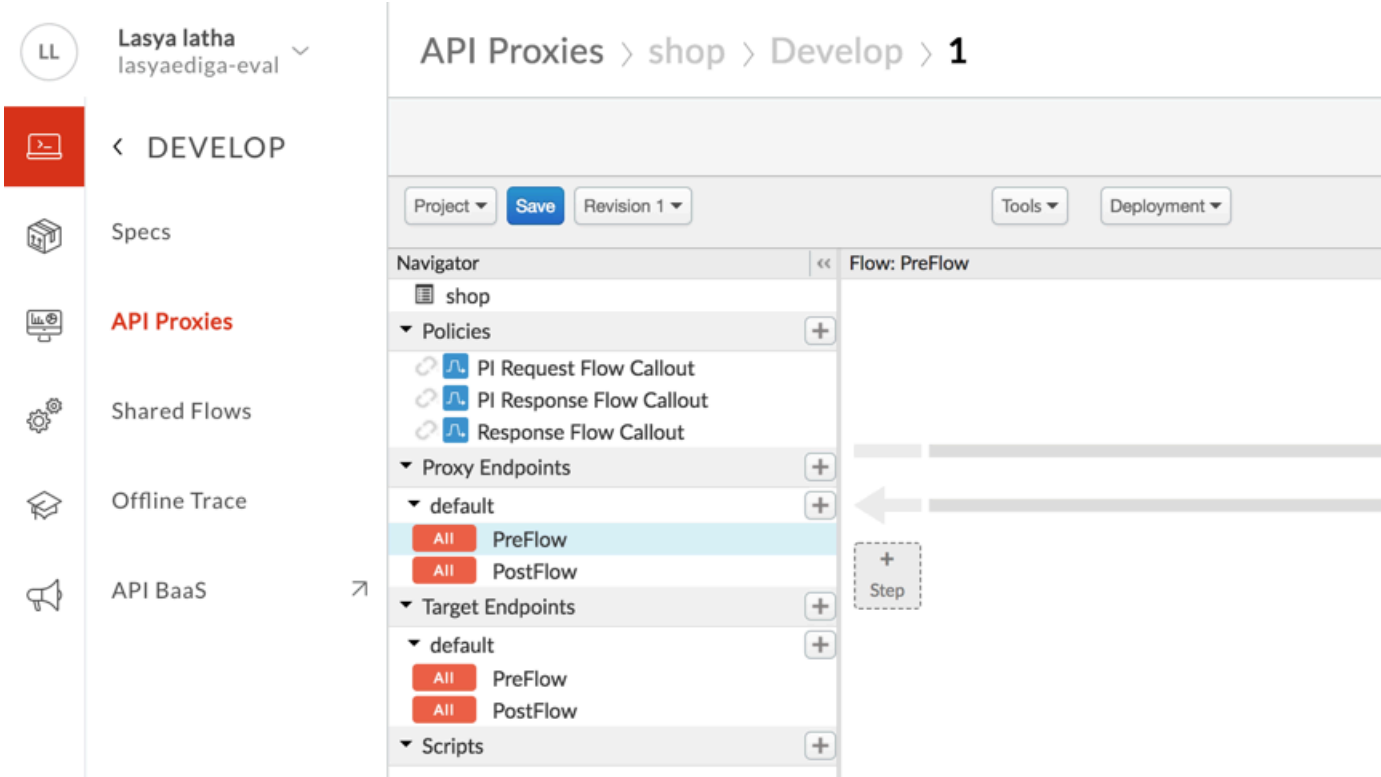


Proxy.

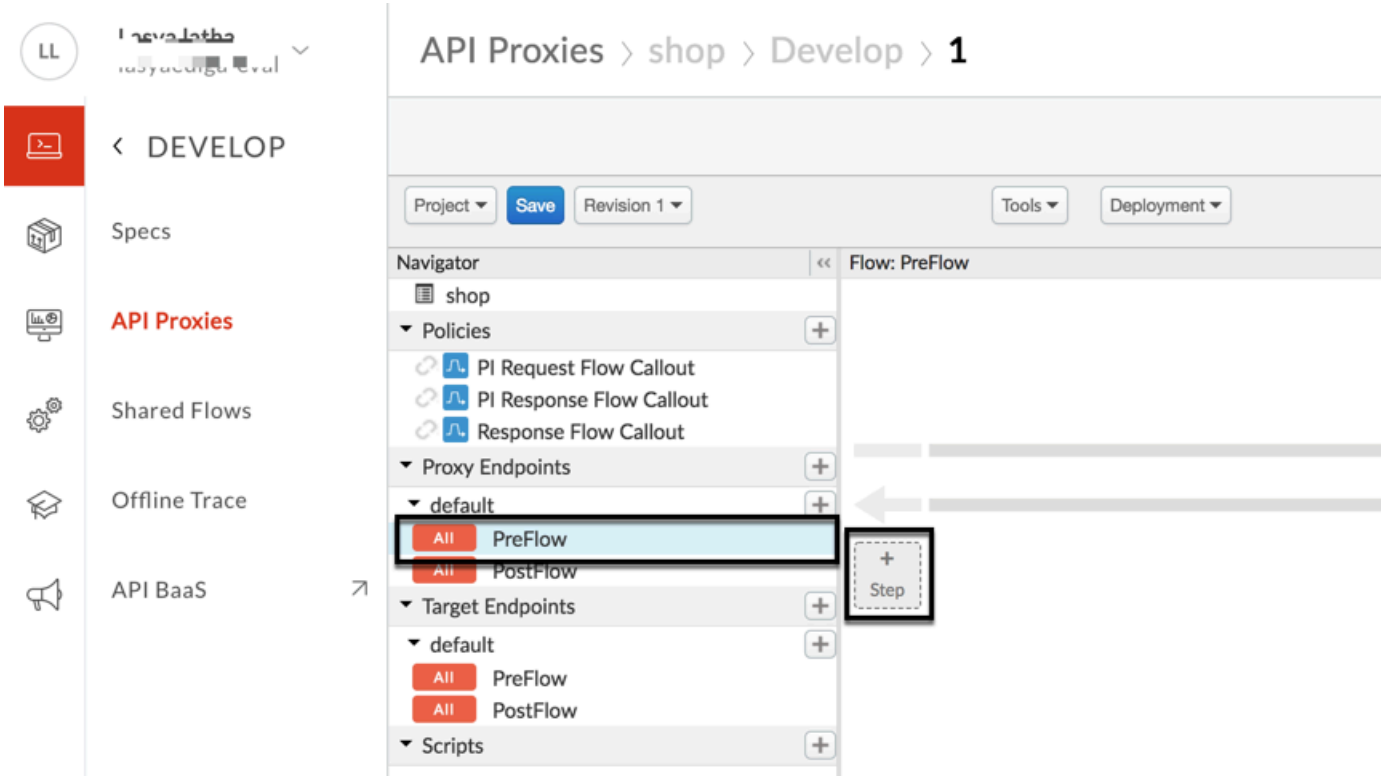
2. Click on the API name on which you want to apply the policy. The Develop page is displayed:



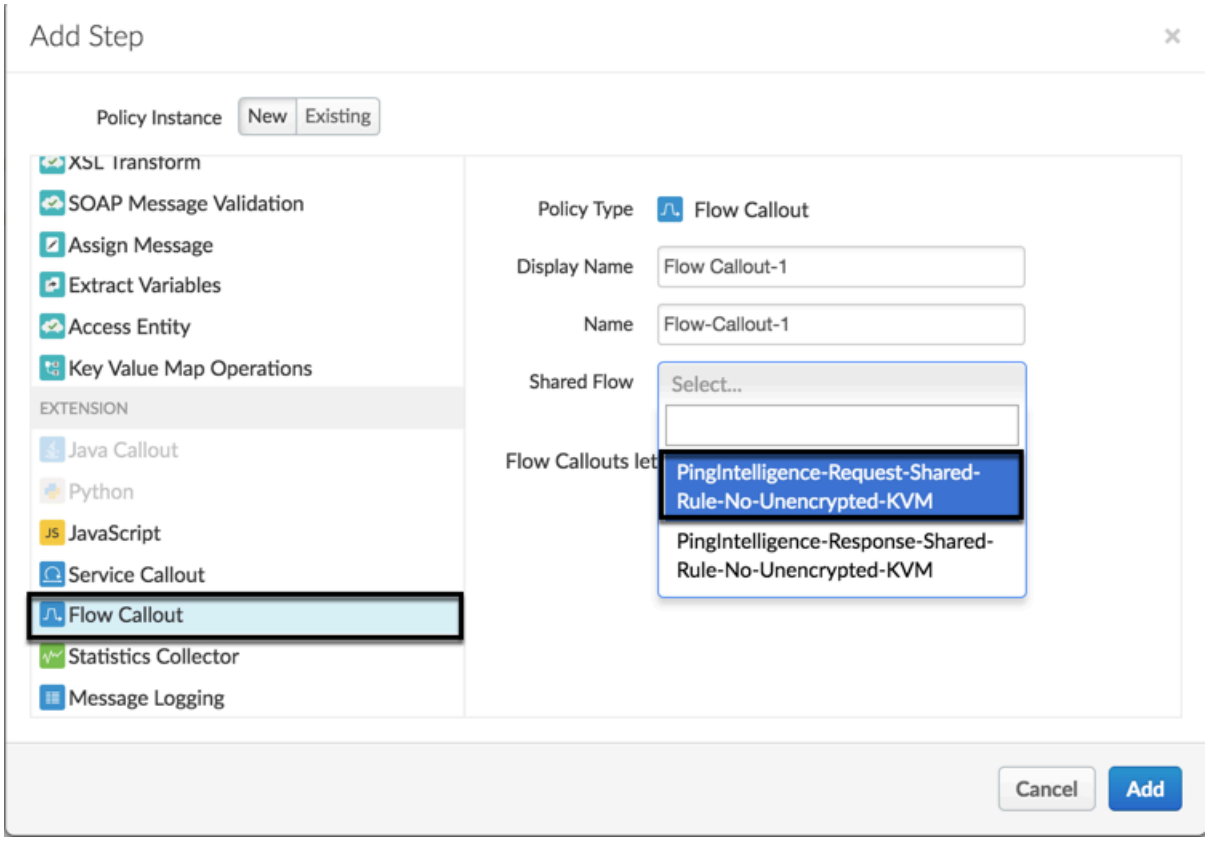
3. On the Develop page, click on the **DEVELOP** tab:



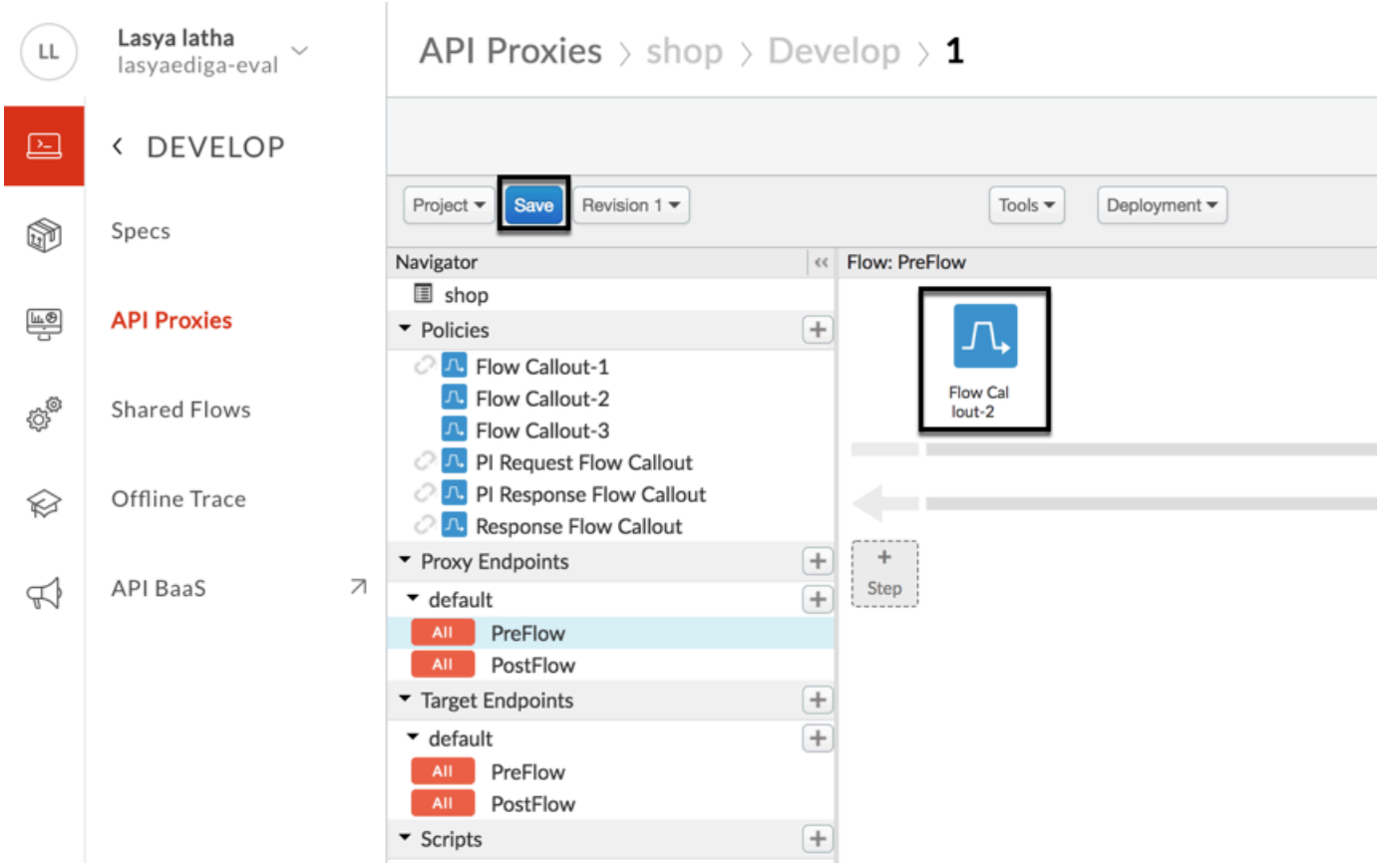
4. In the **DEVELOP** tab, choose **PreFlow** under **Proxy Endpoints**, and click on **+ Step** for request. The Add Step window is displayed:



5. In the Add Step window, select **Flow Callout**. From the **Shared Flow** drop down list, select the Request rule and click on **Add**:



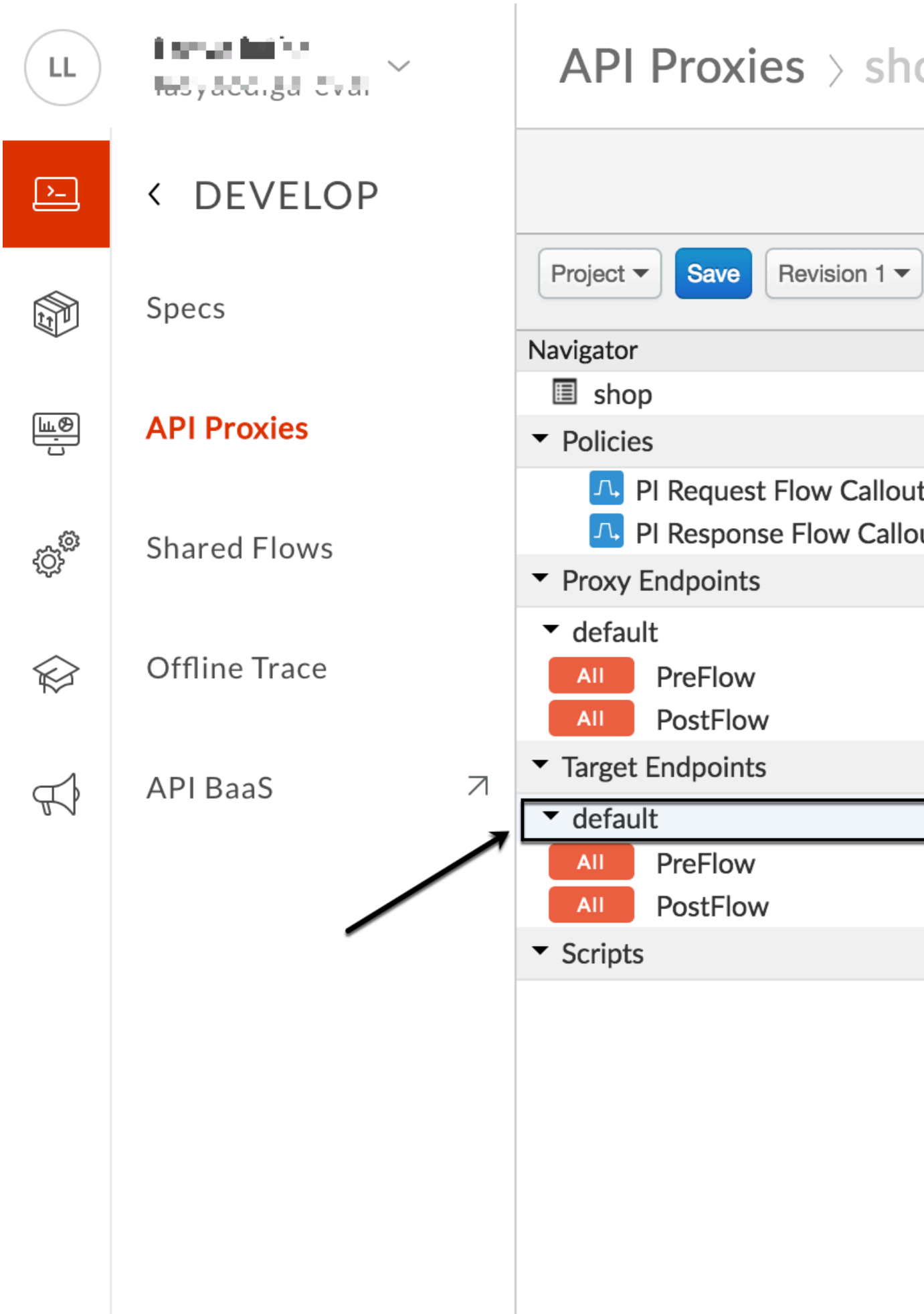
6. Repeat step 5 for Response rule.
7. Request and Response rules are added. Click on Save:



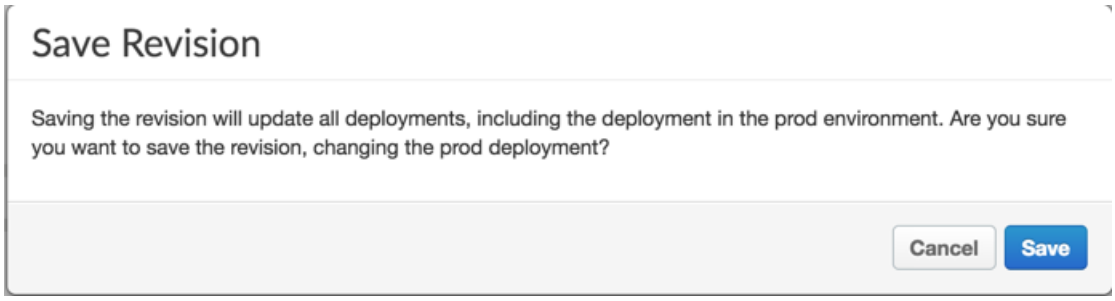
8. Click on **default** and enter the following lines in the <HTTPTargetConnection> tag:

```
<Properties>
            <Property name="success.codes">1xx,2xx,3xx,4xx,5xx</Property>
```

```
</Properties>
```



9. Save the Revision:



Change deployed policy mode

You can change the type of policy deployed from Flow Hook to Flow Call Out or Flow Call Out to Flow Hook using the PingIntelligence policy tool. To change the type of policy complete the following steps:

1. Undeploy the deployed policy by entering one of the following command based on the policy and certificate used:
   - **Undeploy a Flow Hook policy using 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 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
```

- **Undeploy a Flow Call Out policy using 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 Call Out policy using 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
```

2. Deploy the other policy by following the steps detailed for *Flow Hook* or *Flow Call Out*

ⓘ **Note:** Using the above steps you can also change the use of security certificate from self-signed to CA-signed or from CA-signed to self-signed.

Add APIs to ASE

After the policy has been deployed to Apigee using the PingIntelligence automated policy tool, add APIs to ASE. Read the following topics to define and add APIs to ASE:

- *API naming guidelines* on page 153
- *Define an Inline API JSON configuration file* on page 194

For more information on ASE sideband deployment, see *Sideband ASE* on page 144.

AWS API gateway integration

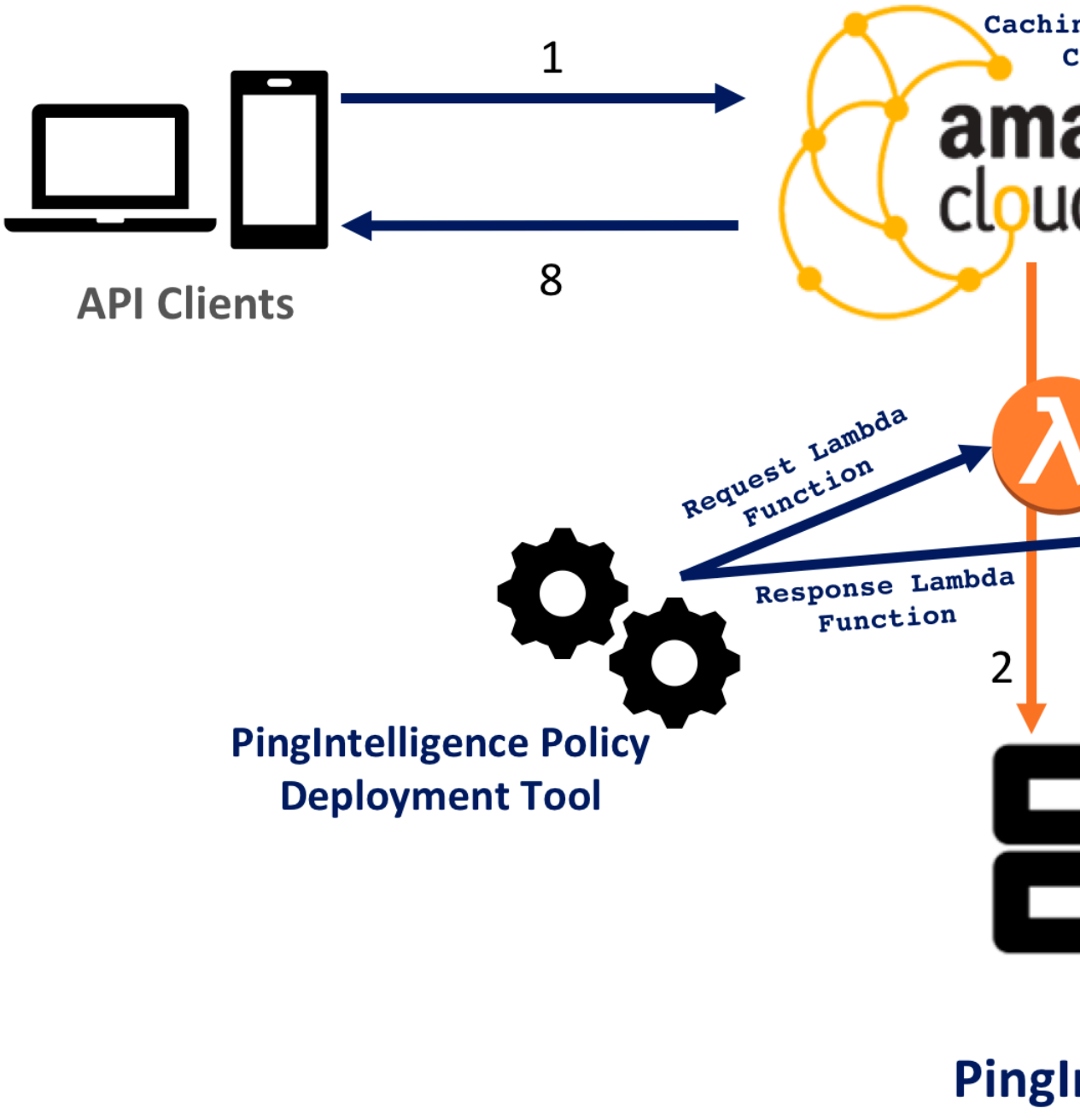
PingIntelligence AWS API Gateway Integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with AWS API Gateway via CloudFront. A PingIntelligence policy is installed in CloudFront and uses Lambda functions to pass API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.

PingIntelligence provides an automated tool to deploy a PingIntelligence policy which is implemented using the AWS Lambda functions. The policy requires AWS CloudFront to be present with all types of caching disabled. 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 newly created instance.

ⓘ **Important:** Up to 1000 QPS, the default Lambda memory value is sufficient. (See the *aws.properties* file for default origin response value). For a larger QPS, contact the PingIntelligence team.

The following diagram shows the logical setup of PingIntelligence ASE and



CloudFront:

Here is the traffic flow through the CloudFront and PingIntelligence for APIs components.

1. 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 PingIntelligence AI engine generated Blacklist. If all checks pass, ASE returns a 200-OK response to the AWS Lambda. If not, a different response code (403) is sent to AWS Lambda. The request information is also logged by ASE and sent to the AI Engine for processing.

4. If CloudFront receives a 200-OK response from ASE, then it forwards the client request to the backend server. Otherwise, the CloudFront blocks the client when blocking is enabled for the API.
5. The response from the backend server is received by CloudFront.
6. The Lambda response function 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 AWS Lambda. The response information is also logged by ASE and sent to the AI Engine for processing.
8. CloudFront sends the response received from the backend server to the client.

Prerequisites

Complete the following before running the PingIntelligence AWS policy tool.

Prerequisite:

- Install OpenJDK 11 on the system running the PingIntelligence policy tool.
- **Install PingIntelligence software**

  PingIntelligence should be installed and configured. Refer to the PingIntelligence deployment guide for your environment.
- **AWS admin account**: To deploy the PingIntelligence sideband policy, an AWS admin account is required.

> ⓘ **Note:** Make sure that AWS cross-account is **not** used to deploy PingIntelligence policy.

- **Update CloudFront configuration**: Verify the following options are configured correctly:
  - **Disable Caching**: 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.
  - **TTL**: Confirm that **Minimum TTL**, **Maximum TTL**, and the **Default TTL** are set to 0
  - **Forward Cookies**: Select **All** from the drop-down list
  - **Query String Forwarding and Caching**: Select **Forward all, cache based on all** from the drop-down list

Edit Behavior

Allowed HTTP Methods
- ○ GET, HEAD
- ○ GET, HEAD, OPTIONS
- ◉ GET, HEAD, OPTIONS, PUT, POST, PATCH

Field-level Encryption Config

Cached HTTP Methods
GET, HEAD (Cached by default)
- ☐ OPTIONS

Cache Based on Selected Request Headers
None (Improves Caching)

Learn More

Object Caching
- ○ Use Origin Cache Headers
- ◉ Customize

Learn More

Minimum TTL 0

Maximum TTL 0

Default TTL 0

Forward Cookies All

Query String Forwarding and Caching Forward all, cache based on all

Smooth Streaming
- ○ Yes
- ◉ No

Restrict Viewer Access (Use Signed URLs or Signed Cookies)
- ○ Yes
- ◉ No

- **Lambda function**: 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.
- **Verify that ASE is in sideband mode**

  Check if ASE is in `sideband` mode by running the following command in the 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                       : 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.
- **Enable sideband authentication:** 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
  ```
- **Generate sideband authentication token**

  A token is required for CloudFront to authenticate with ASE. This token is generated in ASE and configured in the `aws.properties` file of PingIntelligence automated policy tool. To generate the token in ASE, enter the following command in the ASE command line:

  ```
  # ./bin/cli.sh -u admin -p admin create_sideband_token
  ```

  Save the generated authentication token for further use.

Configure automated policy tool

Download

Complete the following steps to download and install the PingIntelligence policy tool:

1. *Download* the PingIntelligence policy tool to the `/opt` directory.
2. Complete the following steps to untar the policy tool:
   a. At the command prompt, type the following command to untar the policy tool file:

   ```
   tar –zxvf <filename>
   ```

   For example:

   ```
   tar –zxvf pi-aws-4.0.tar.gz
   ```

   b. To verify that the tool successfully installed, type 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 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.

Configure the automated tool

Configure the `aws.properties` file available in the `/pingidentity/pi/aws/config/` directory. 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 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 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. 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 SSL communication between Lambda functions and ASE. The default value is `true`.
`ase_sideband_token`	Enter the ASE token generated during the *prerequisite* step.

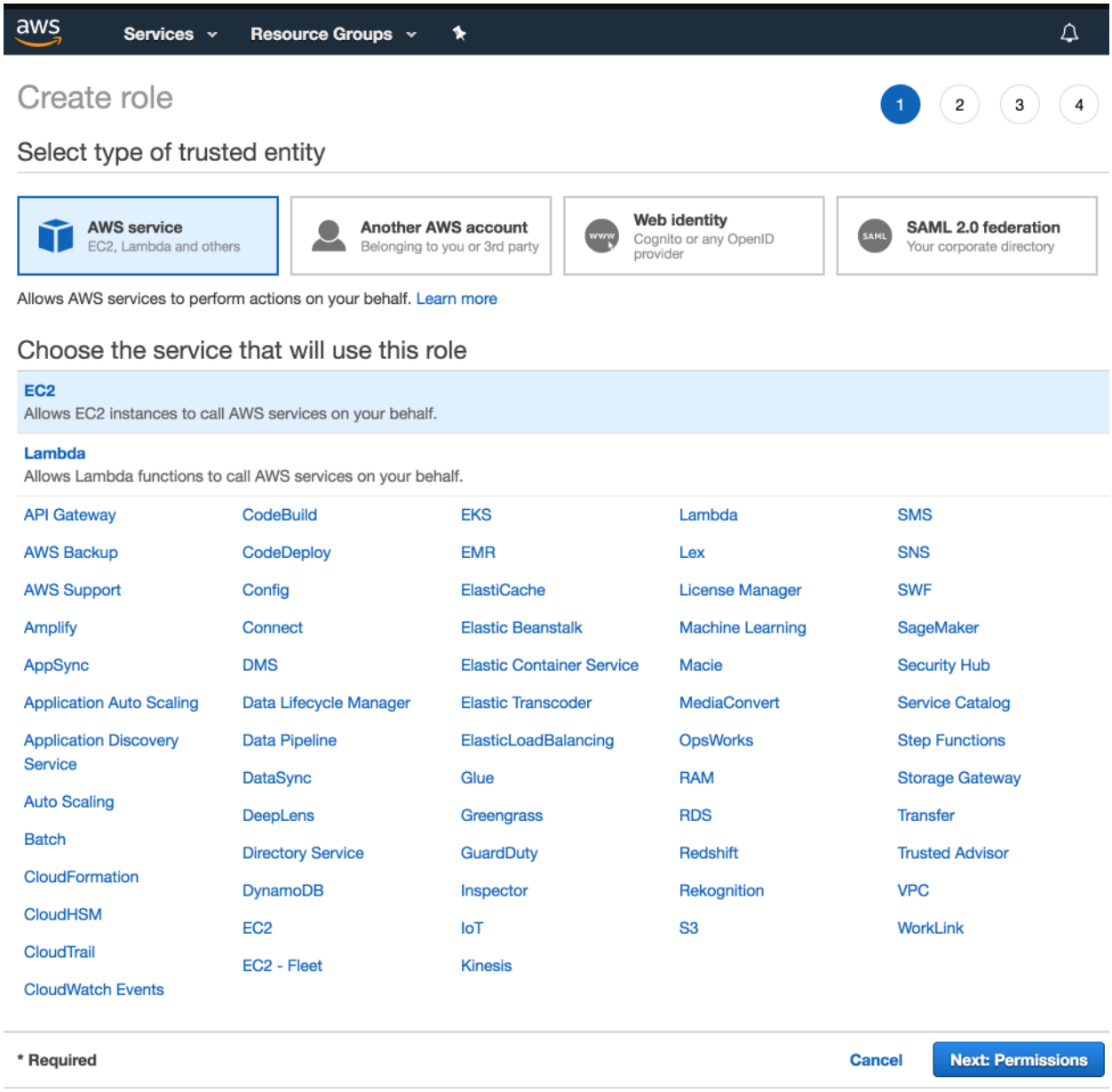
Following is a sample `aws.properties` file:

```
# 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
```

Create Role

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. Complete the following steps to create and configure.

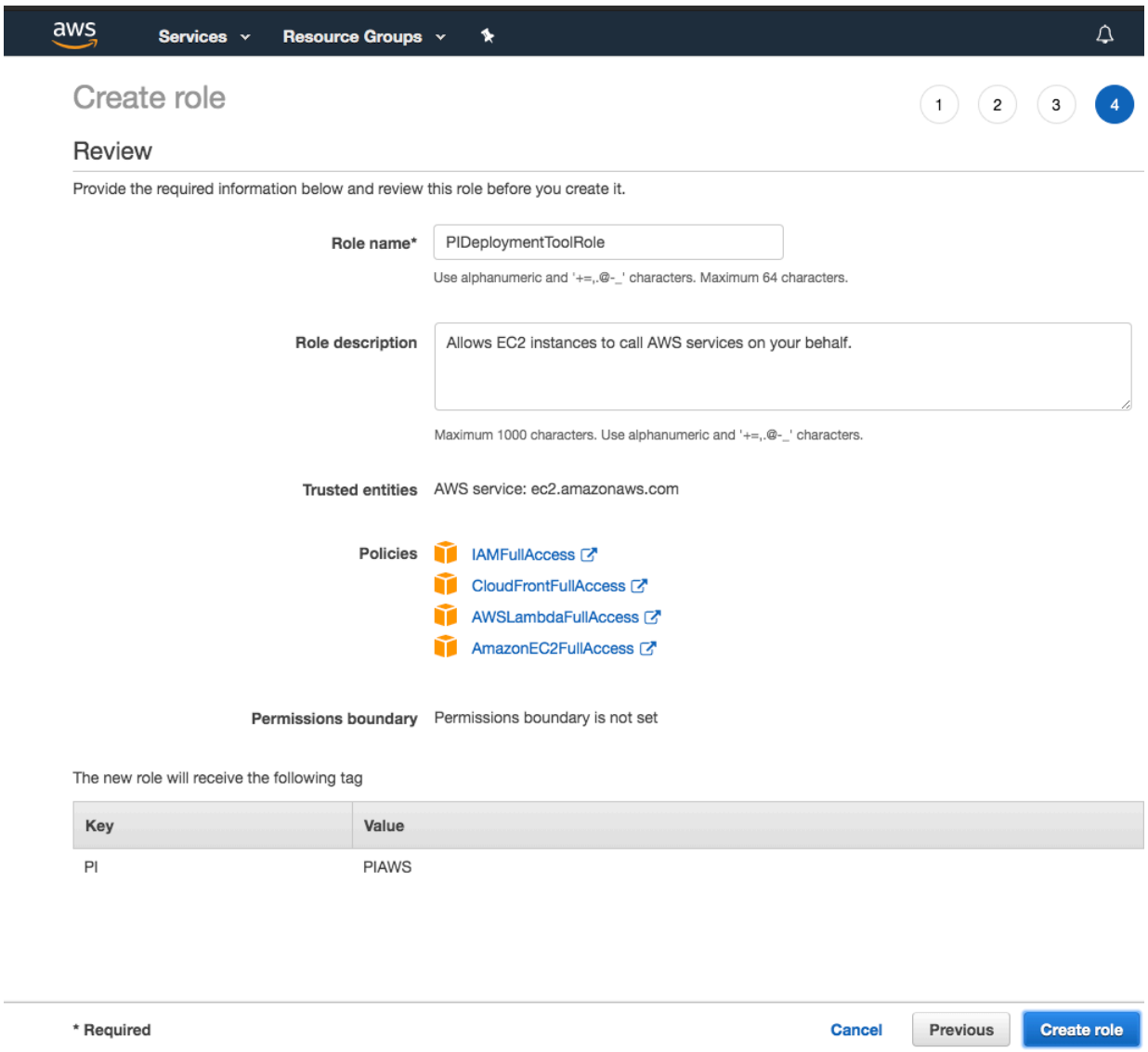
1. Select EC2 as service and click on **Next: Permissions** button:



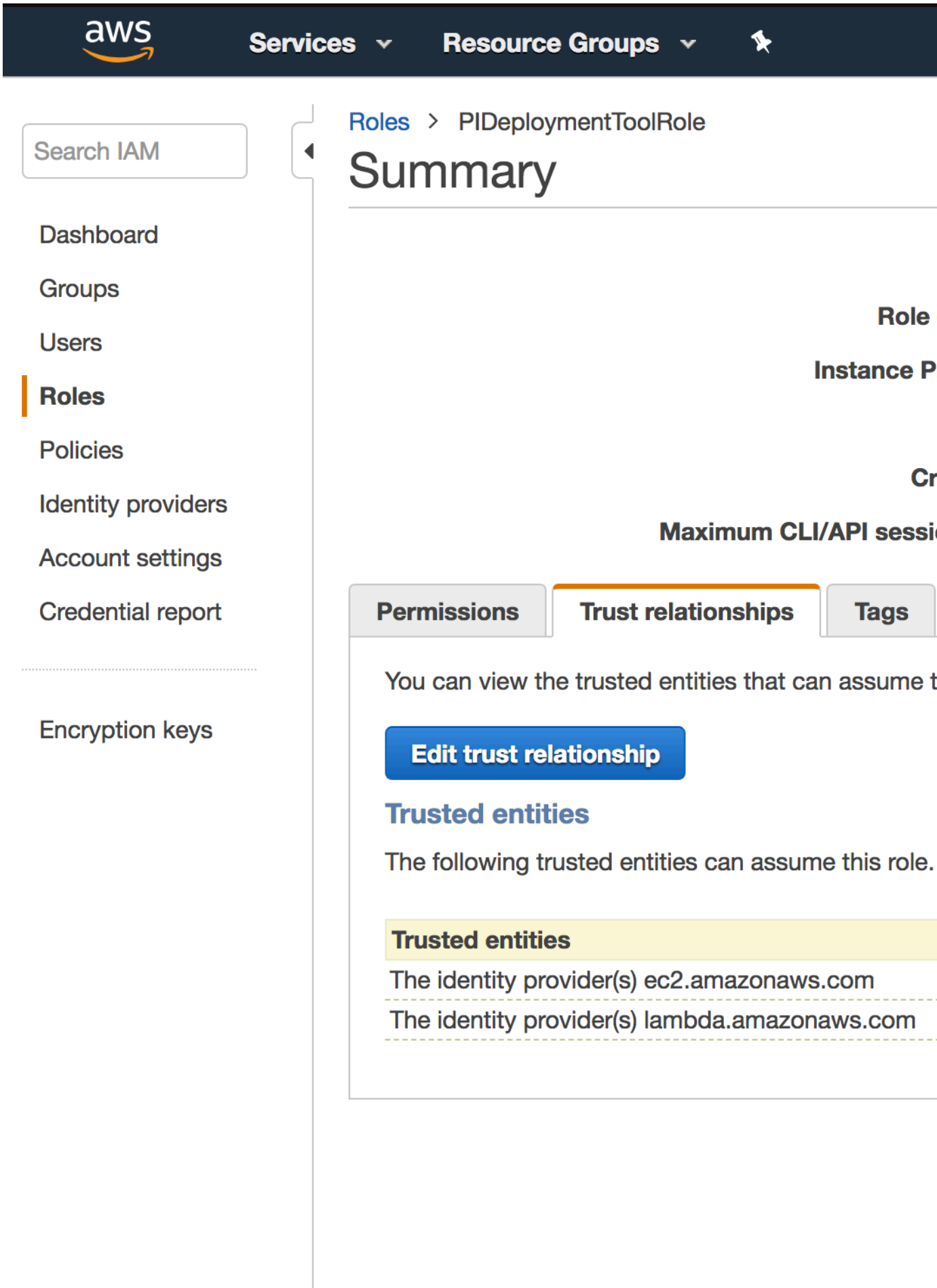
2. Choose the following three Policies and provide a name for each role (for example, PIDeploymentToolRole):

   - `IAMFullAccess`
   - `AWSLambdaFullAccess`
   - `CloudFrontFullAccess`
   - `AmazonEC2FullAccess`

   After providing the name, click on **Create role**.



3. In the Summary page of the role that you created in step 2, click on the **Trust relationships** tab and then click on **Edit trust relationship** button:



4. In the **Edit Trust Relationship** page, enter the following lines and click on **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"
    }
  ]
```

```
}
```

Edit Trust Relationship

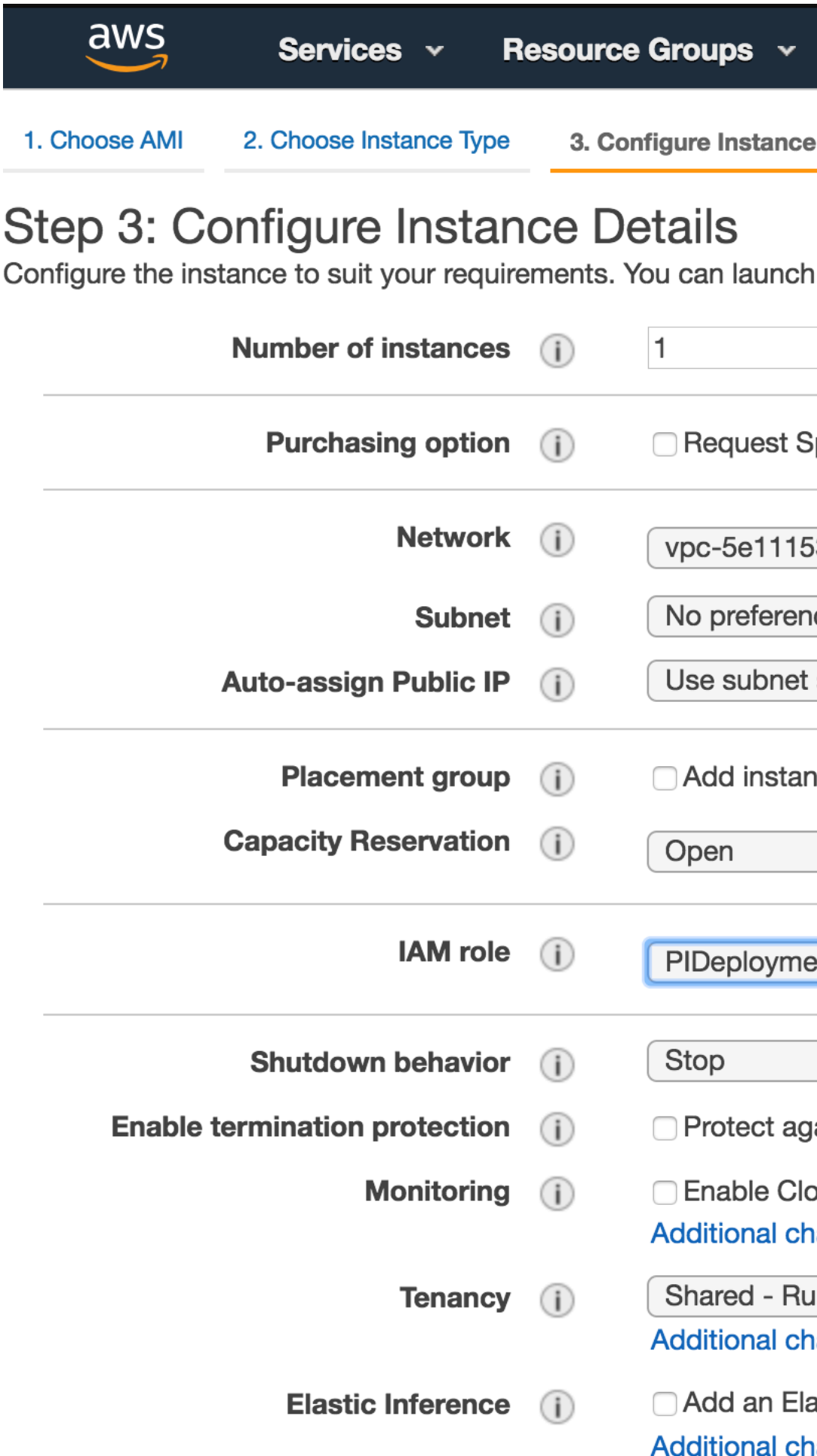
Edit Trust R

You can customize trus

Policy Document

```
1  {
2    "Versio
3    "Statem
4      {
5        "Ef
6        "Pr
7          "
8        },
9        "Ac
10     },
11     {
12       "Ef
13       "Pr
14         "
15       },
16       "Ac
17     }
18   ]
19 }
```

5. Configure the **IAM role**, as the role that you created (for example,



Deploy PingIntelligence Policy for AWS

Using the PingIntelligence AWS policy tool, deploy the PingIntelligence policy in AWS @Lambda in the North Virginia (US-East-1) region. Note: the policy must currently be initially deployed in this region. The Lambda function pushes the PingIntelligence policy to the Amazon CloudFront in the local AWS instances. The PingIntelligence 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.

To deploy the PingIntelligence policy, run the following command:

```
/opt/pingidentity/pi/aws/bin$ deploy.sh -ca

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.
```

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

Check the status of deployment: To check the status of the PingIntelligence policy deployment, run the `status.sh` command:

```
/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.
```

API discovery
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 JSON is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

Next steps - Integrate into your API environment

After the policy deployment is complete, refer to the following topics for next steps:

It is recommended to read the following admin guide topics apart from reading the *ASE* and *ABS* Admin Guides:

- *ASE port information*
- *API naming guidelines*
- Adding APIs to ASE in *Sideband ASE*. You can add individual APIs or you can configure a *global API*.
- *Connect ASE and ABS*

After adding APIs to PingIntelligence, the API model needs to be trained. The training of an API model is executed in the ABS AI engine.. The following topics provide a high level view of the process.

- *Train your API model*
- *Generate and view the REST API reports using Postman*
- *View PingIntelligence for APIs Dashboard*.

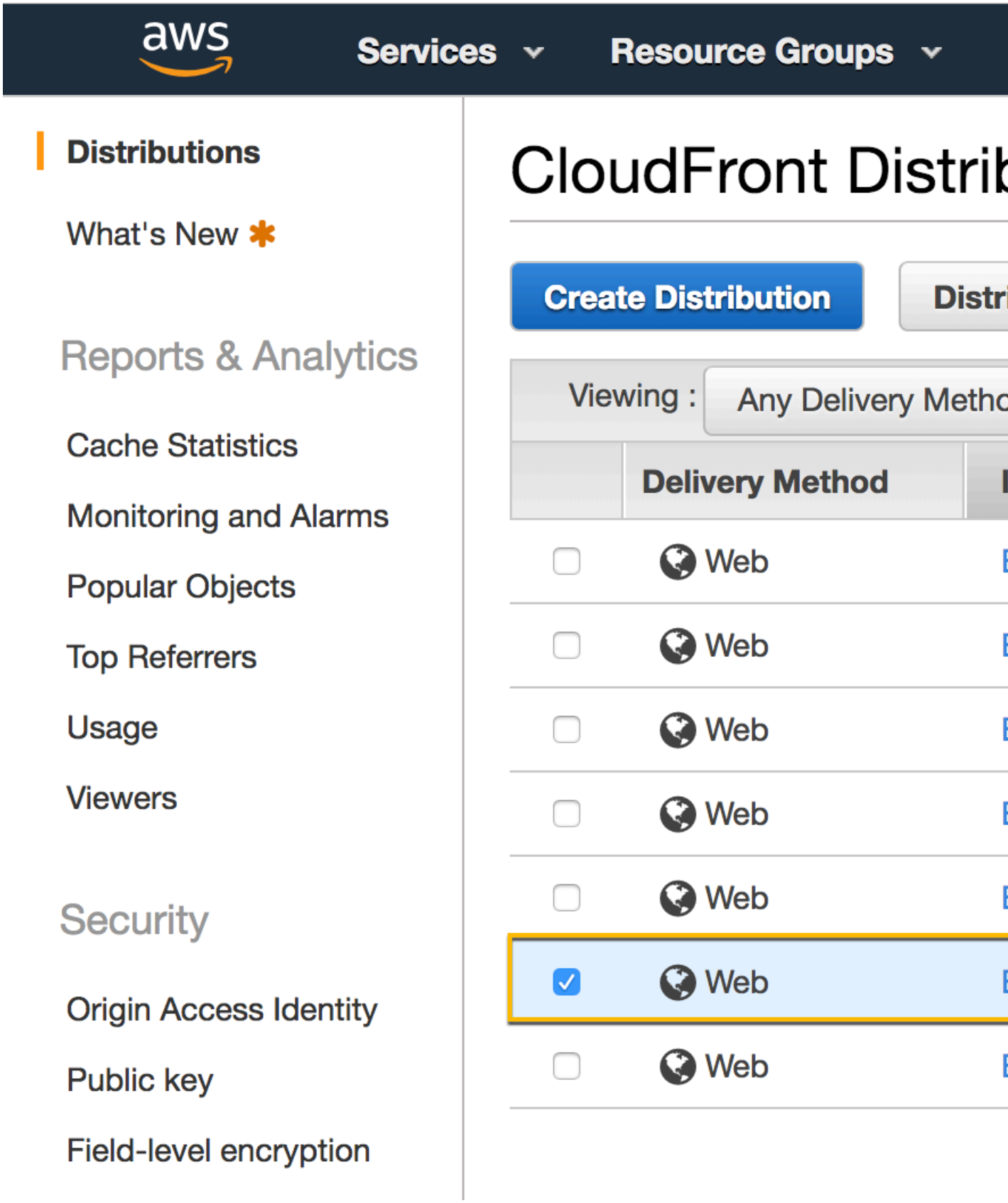
Uninstall CloudFront sideband policy

Remove the PingIntelligence AWS policy with the undeploy tool which detaches the policy from CloudFront. The amount of time required to detach the policy from CloudFront varies depending on the CloudFront region where the policy is deployed.

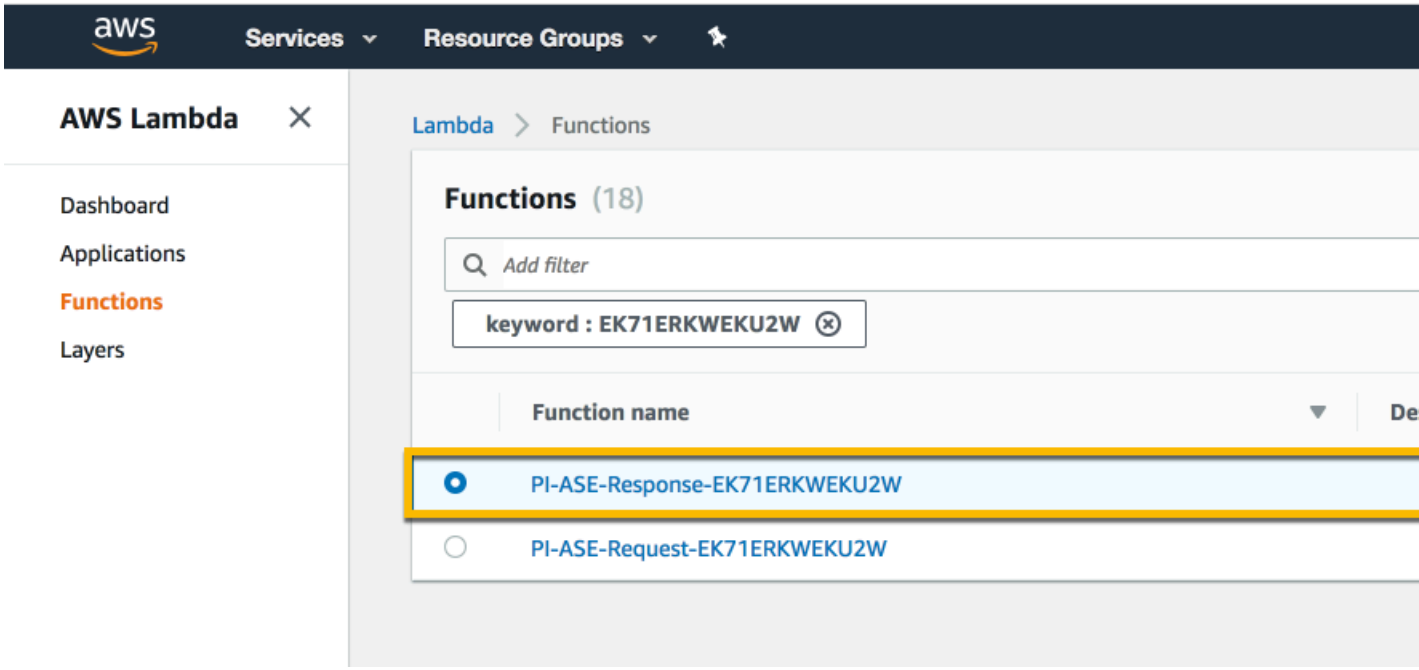
To undeploy the policy, run the following command:

```
/opt/pingidentity/pi/aws/bin$ undeploy.sh
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.
```

Check the progress of detaching the policy from the CloudFront in the AWS GUI as shown in the following screenshot:



After the **State** has moved from Enabled to Disabled, delete the Request and Response Lambda functions. Use the `cloud_front_id` from the `aws.properties` file to search for PingIntelligence Lambda functions.



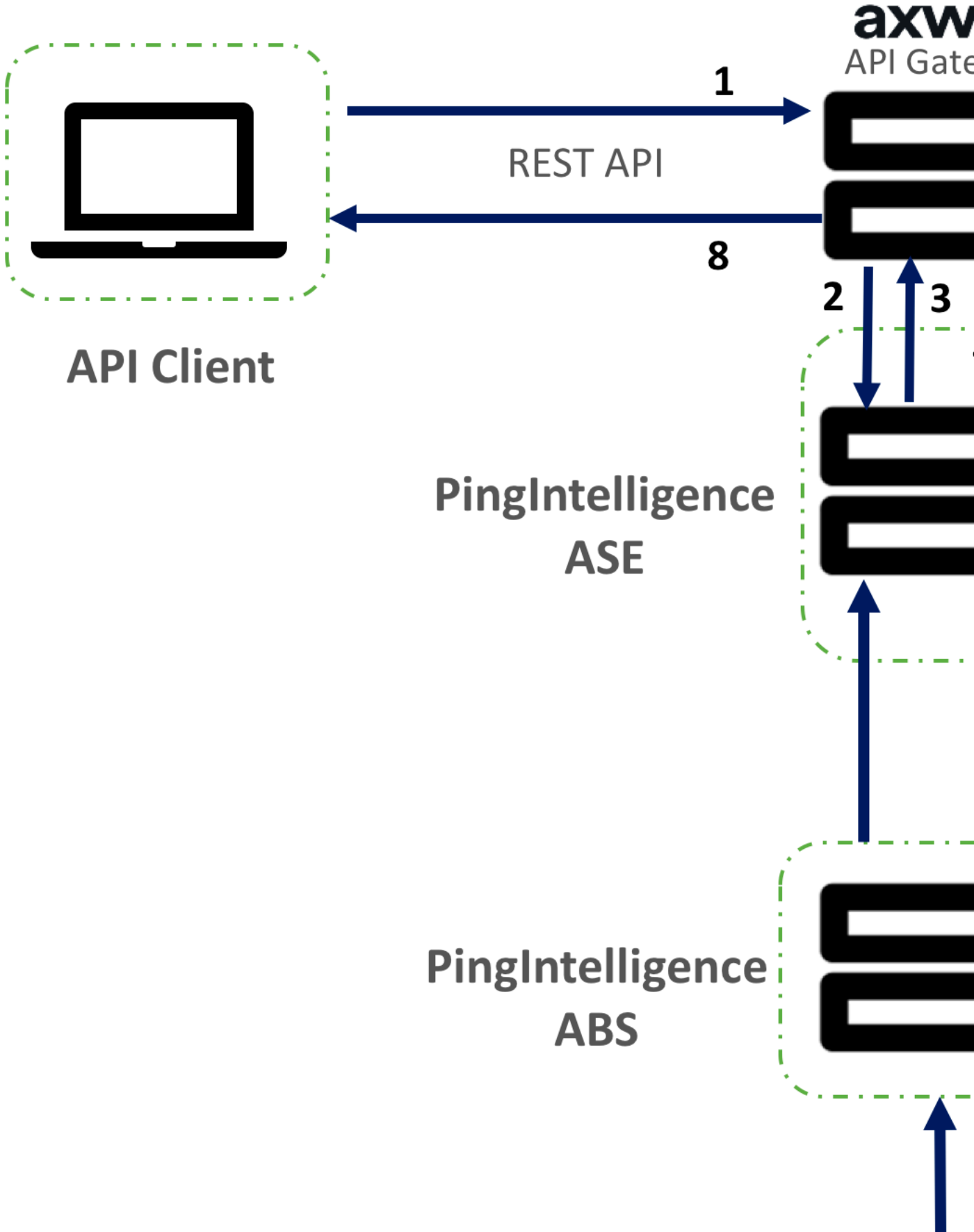
ⓘ **Note:** If the Lambda functions are not deleted, then the following message is displayed on the console: `Deletion of the Lambda function may take up to one hour. Please re-run undeploy.sh after one hour.`

Axway API gateway integration

Axway sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with an Axway API Gateway. 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:



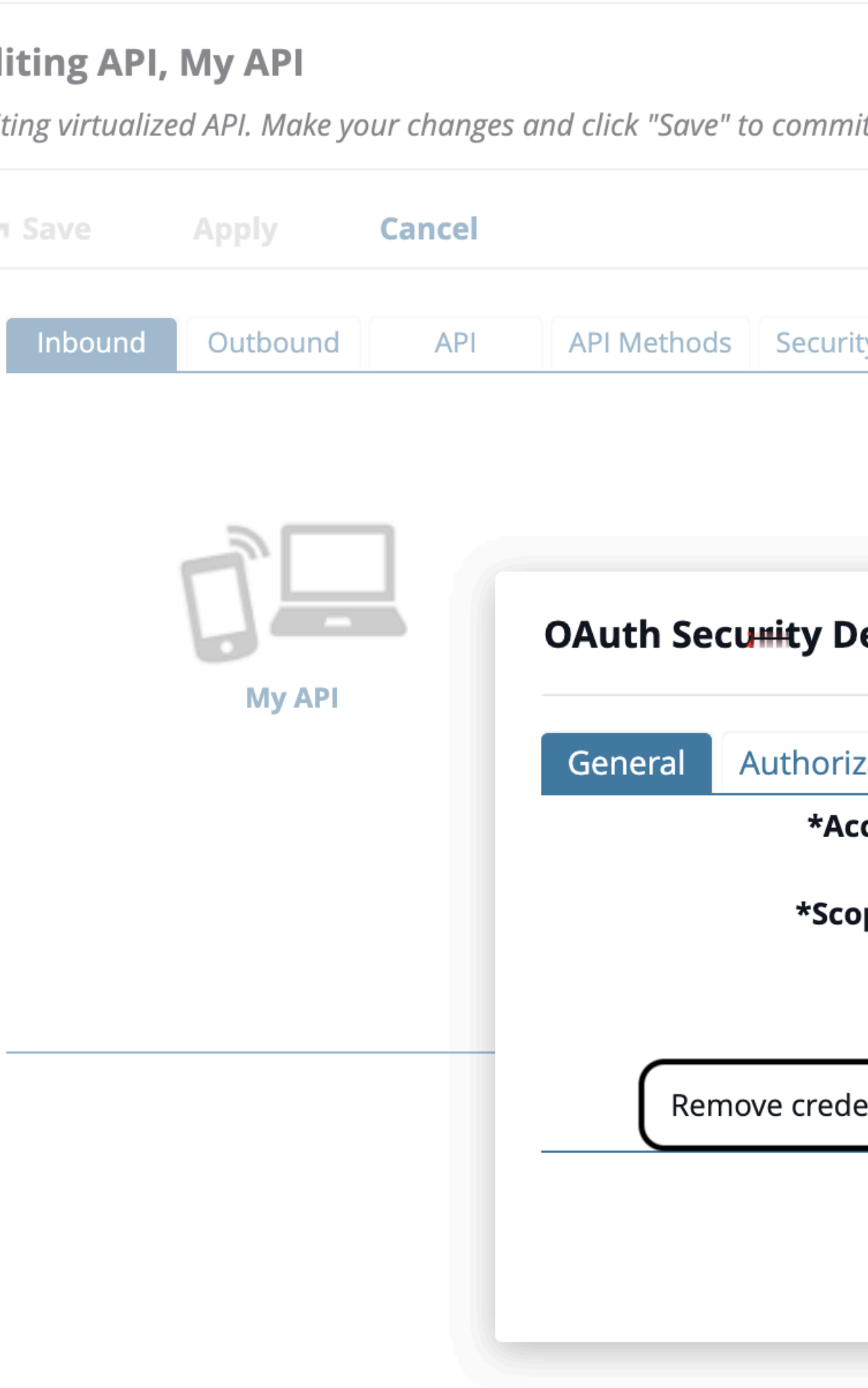
Here is the traffic flow through Axway and PingIntelligence for APIs components.

1. Client sends an incoming request to Axway
2. Axway makes an API call to send the request metadata to 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 AI engine generated Blacklist. 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.
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.

Prerequisites

Complete the following before configuring the Axway API Gateway:

- **Confirm the Axway version** PingIntelligence 4.0 works with Axway 7.5.3 or



later.

- **OAuth token store**: If you wish to detect username based attacks, make sure that OAuth token store is configured in Axway.
- **Install PingIntelligence software**

  PingIntelligence should be installed and configured. Refer to the PingIntelligence deployment guide for your environment.
- **Verify that ASE is in sideband mode**

  Check that 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.
- **Enable sideband authentication:** 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
```

- **Generate sideband authentication token**

  A token is required for Axway to authenticate with ASE. To generate the token in ASE, enter the following ASE command:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use.
- **Port for AAD**

  If you are using *AAD* to automate API definition updates on PingIntelligence, open the following ports:

  - Open the management port to fetch API definitions from Axway. The default port is 8075.
  - Open port 8010 in ASE for AAD to add API definitions.

To connect PingIntelligence ASE with Axway API Gateway, complete the following steps:

- Import the Axway Policy in Axway Policy Studio
- Deploy the Axway Policy
- Import the APIs from the Management VM to Axway API Manager.

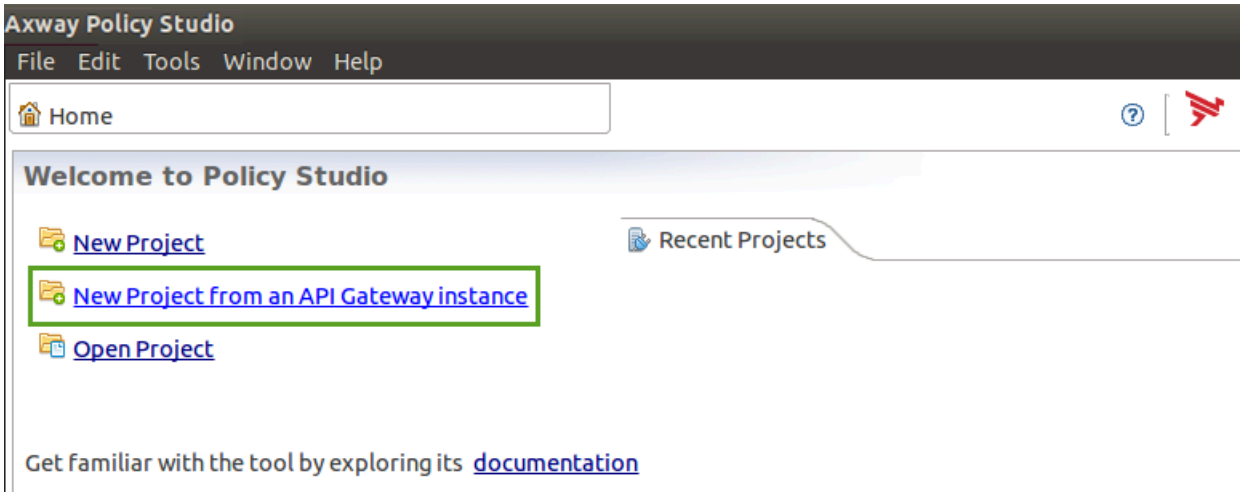
Deploy PingIntelligence policy

Deploying PingIntelligence policy requires completing the following two parts:

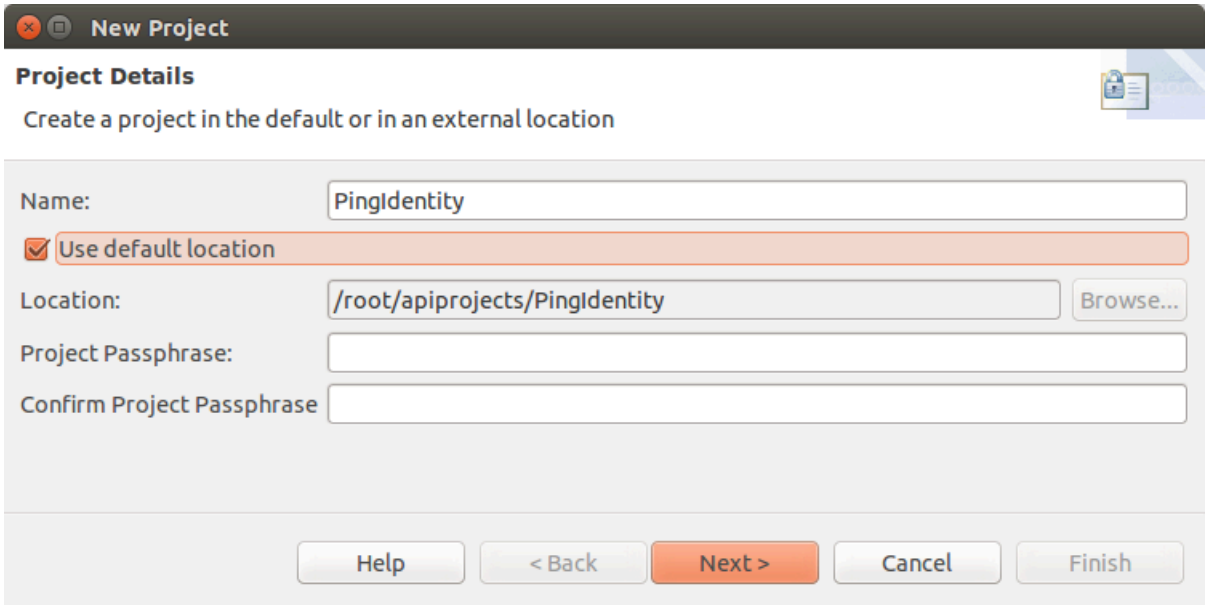
- *Configuring Axway Policy Studio*
- *Configure persistent connection for ASE keep-alive*
- *Configuring Axway API Manager*

Axway Policy Studio configuration

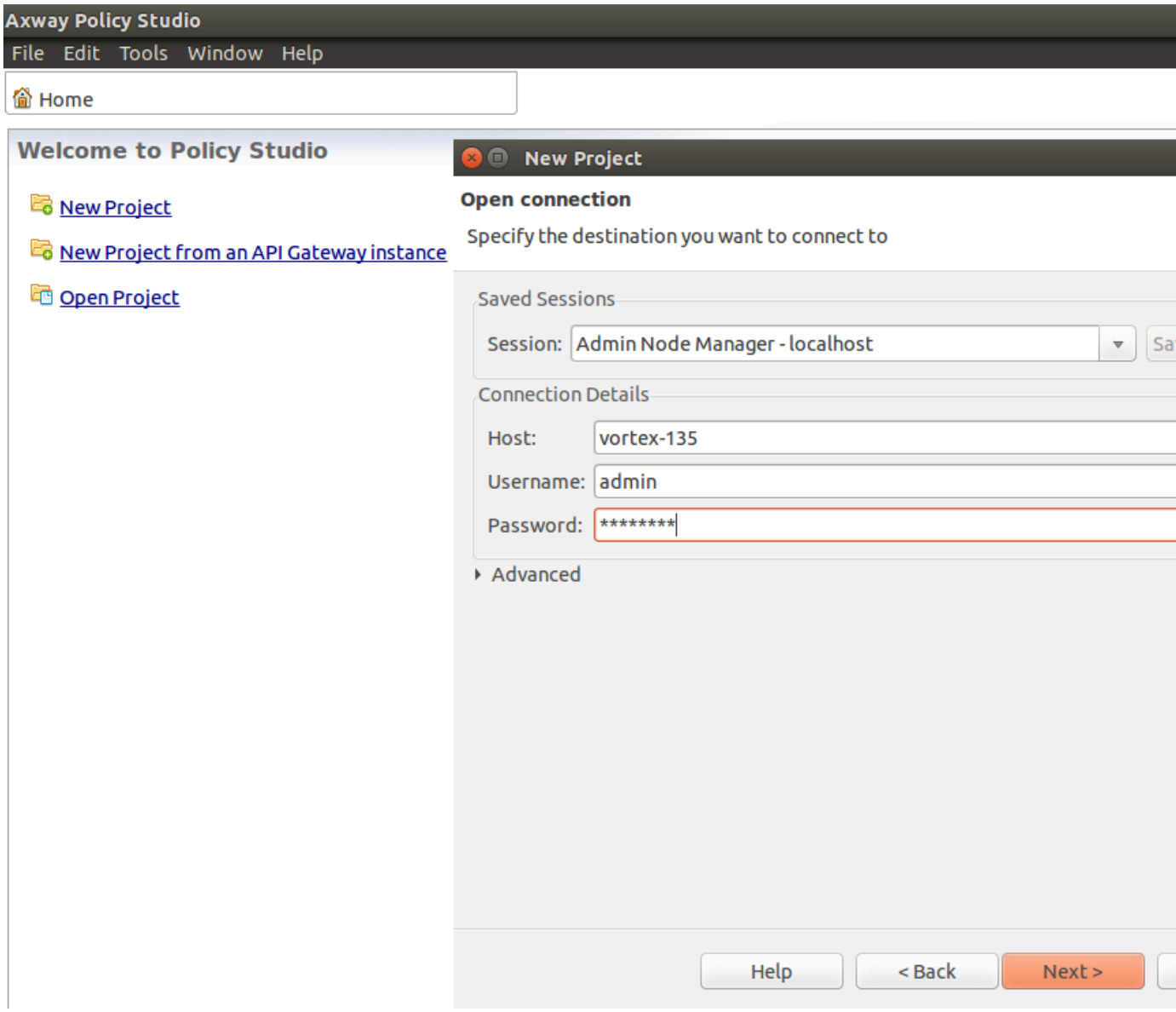
1. Launch Axway Policy Studio and create a new project from an **API Gateway instance**:



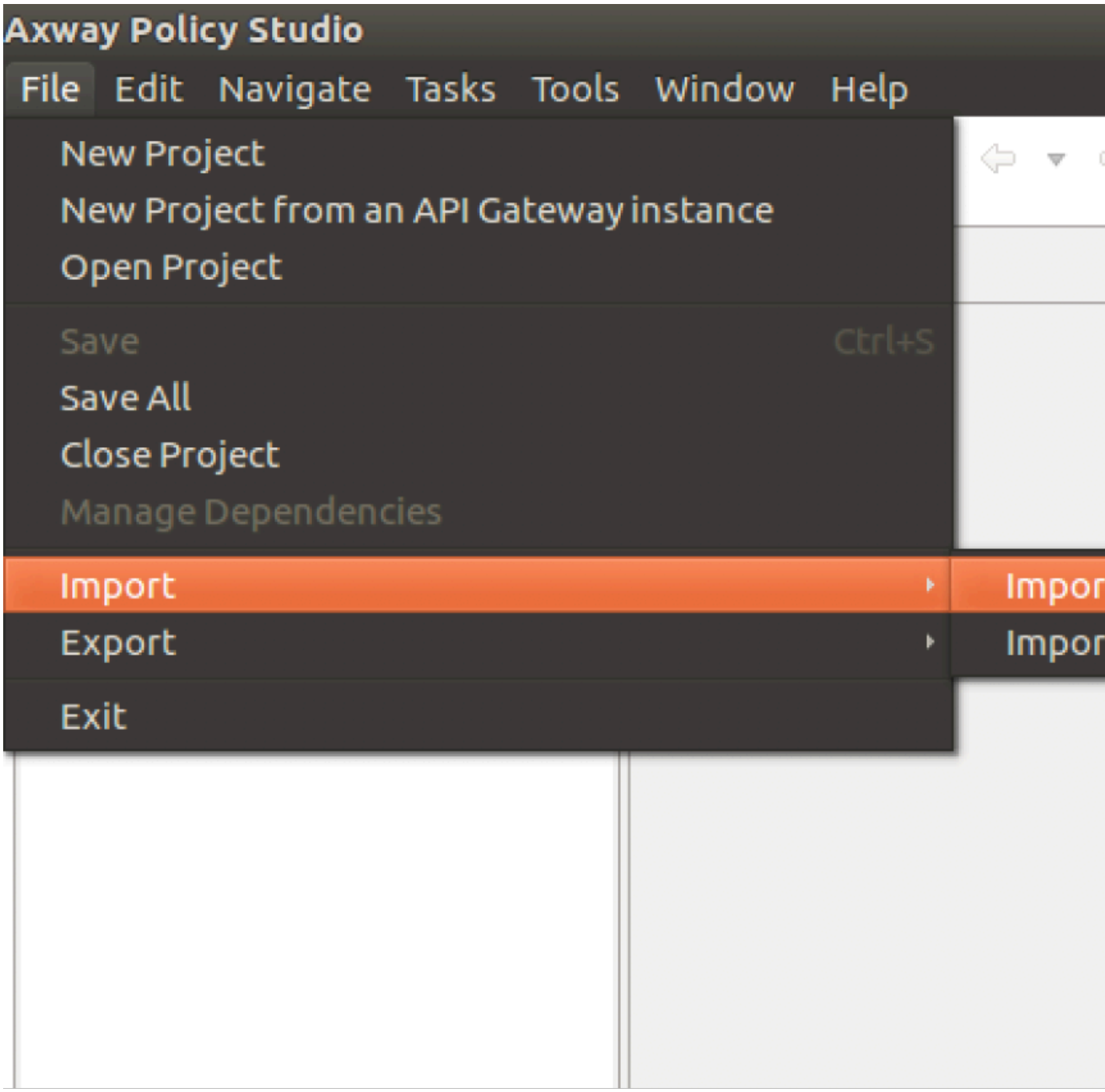
2. In the **New Project** pop-up window, enter the details and click **Next:**



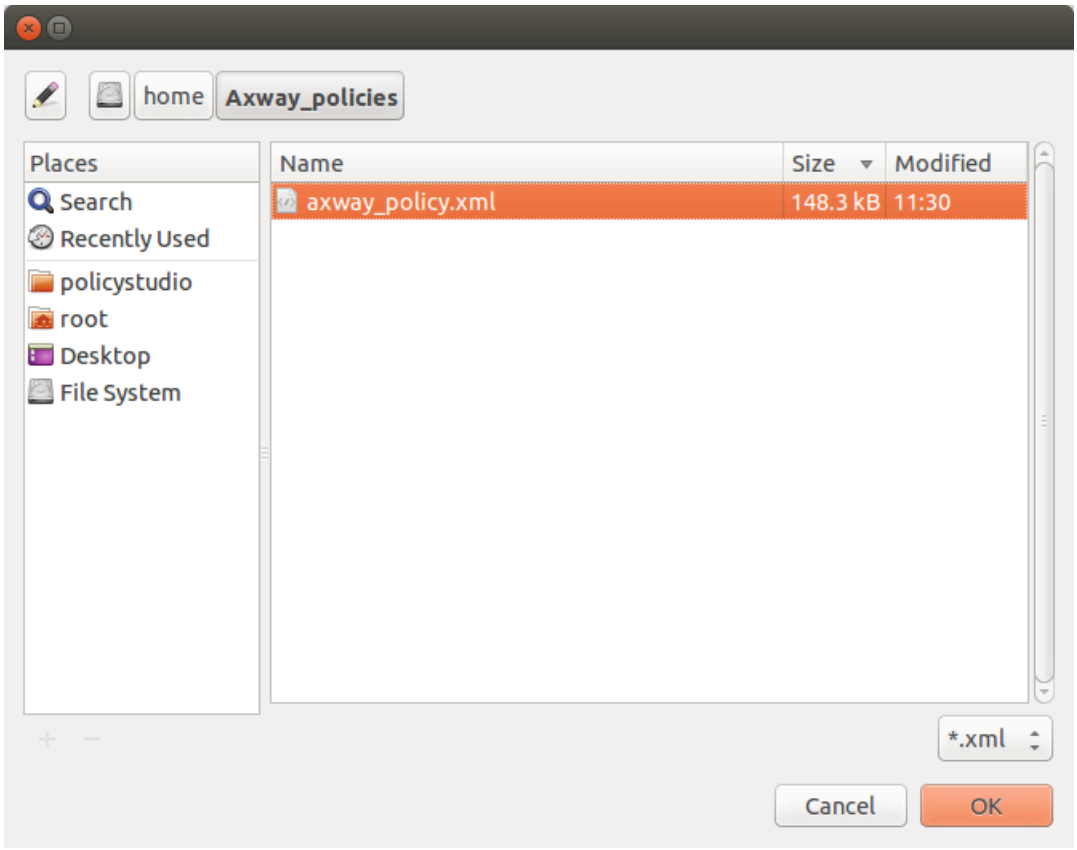
3. Enter **Host** details, **Username**, and **Password** of the API Gateway to connect and click **Next**:



4. Click **Import configuration fragment** from the **File** sub menu in the menu bar

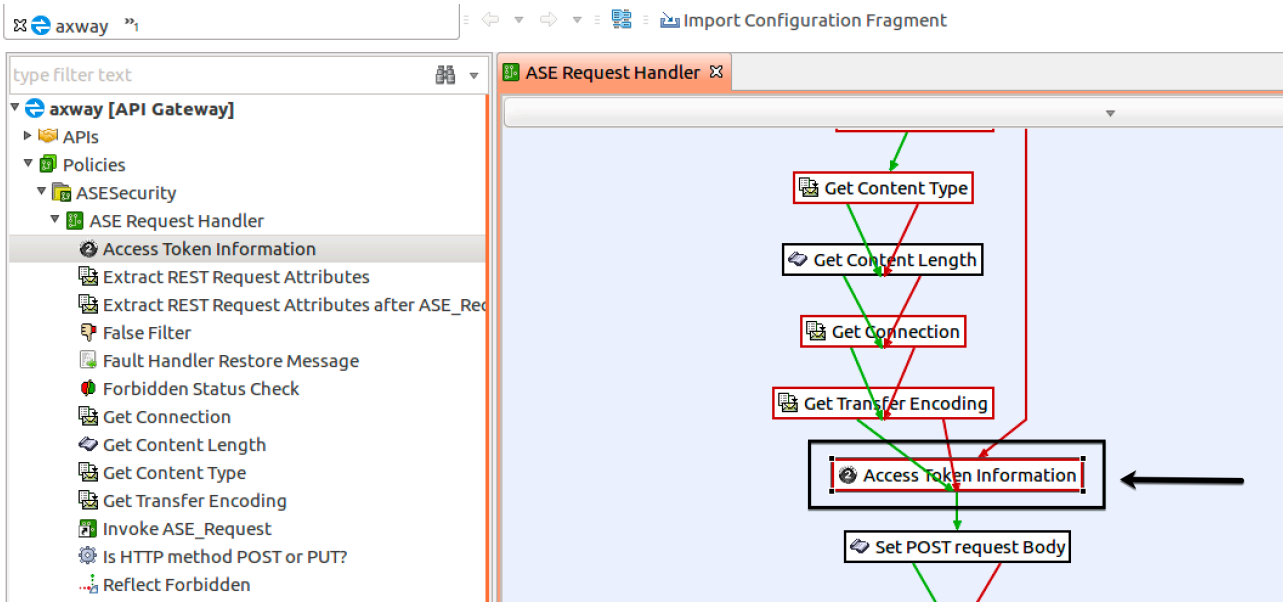


From the pop-up window, import the Axway Policy from the directory where it was saved. Select the policy and click **OK**:

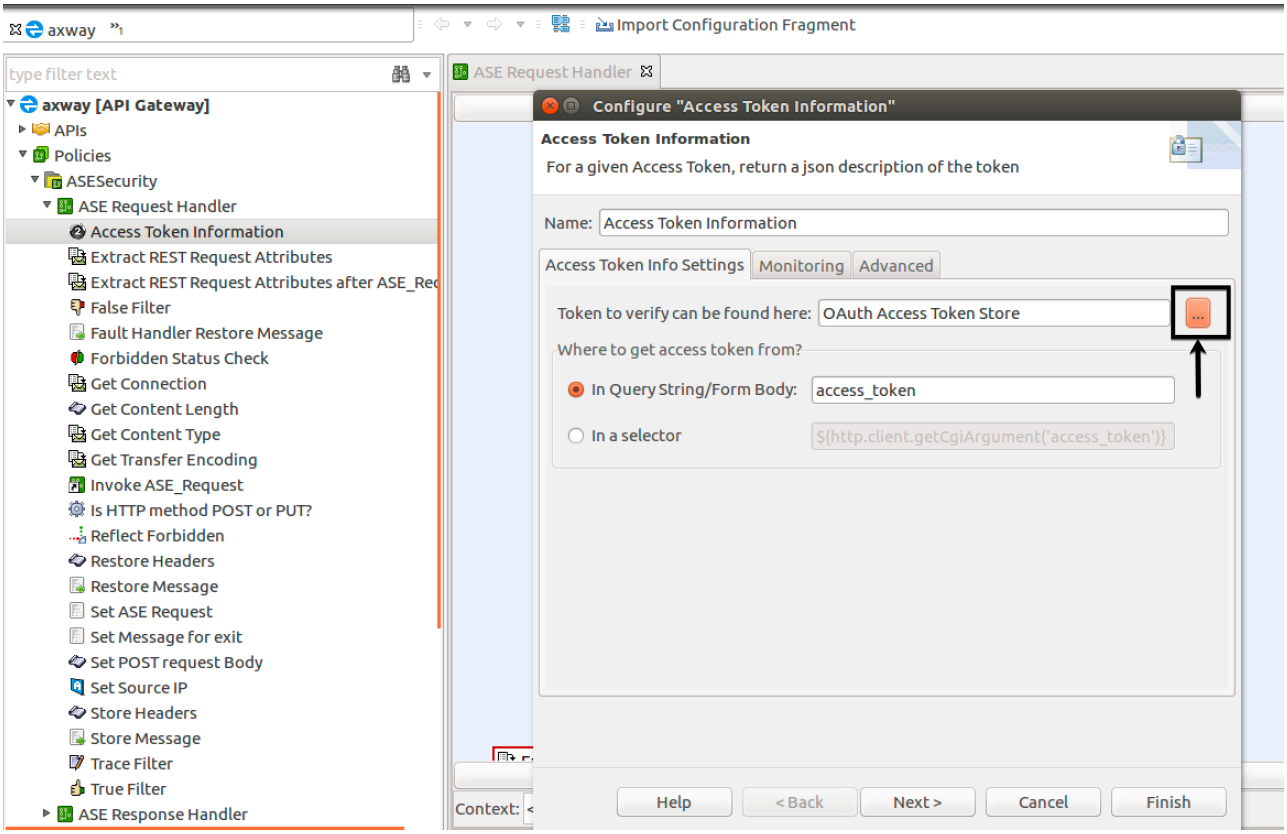


5. After the Axway Policy is imported, click on **Policies** > **ASESecurity** > **ASE Request Handler** > **Access Token Information**. Double click on

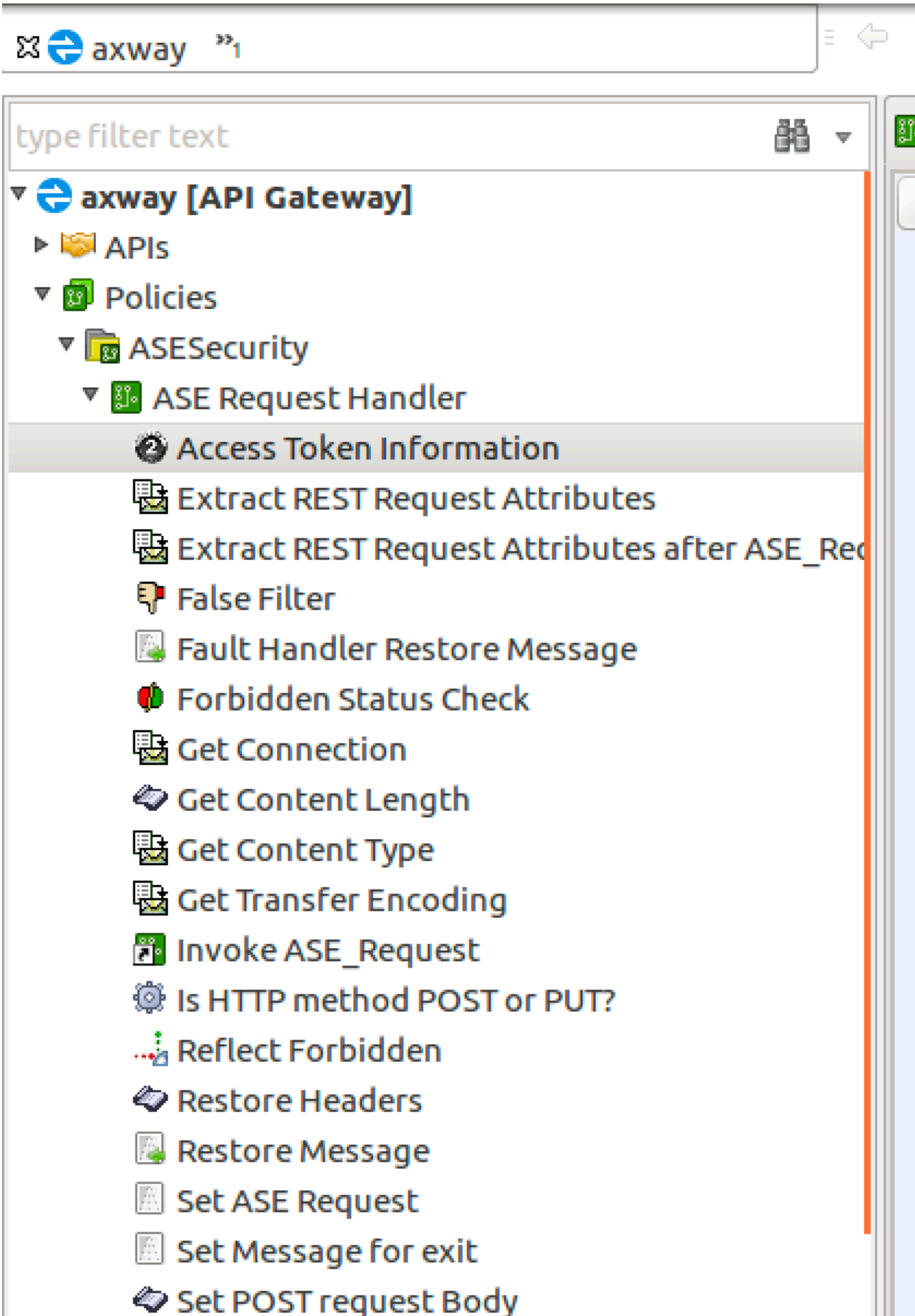
Access Token Information box in the ASE Request Handler window.



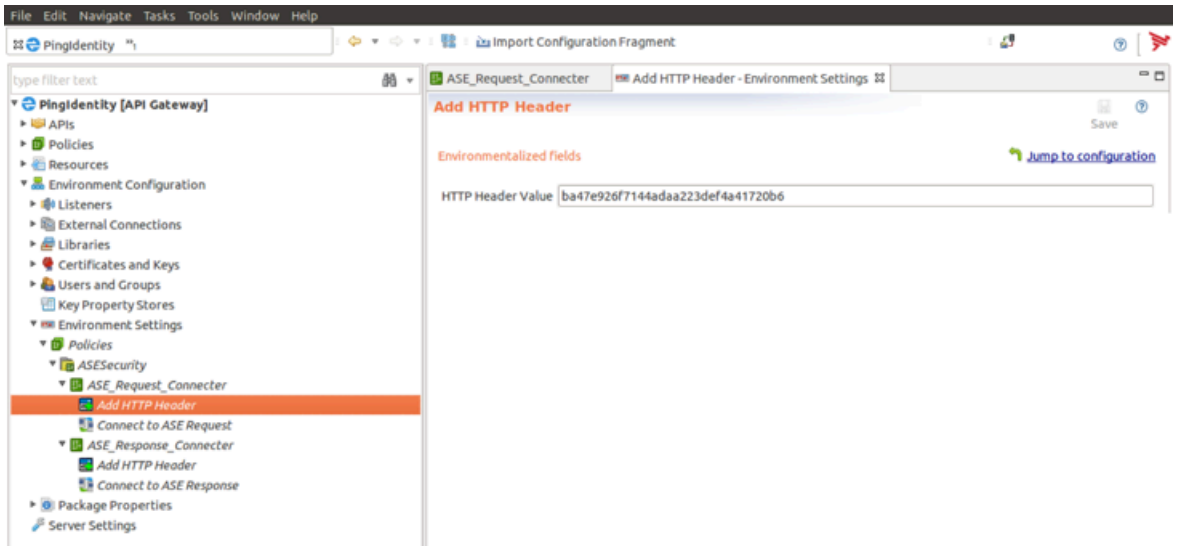
a. In the **Configure "Access Token Information"** pop-up window, enter your OAuth token store information and click the ... button.



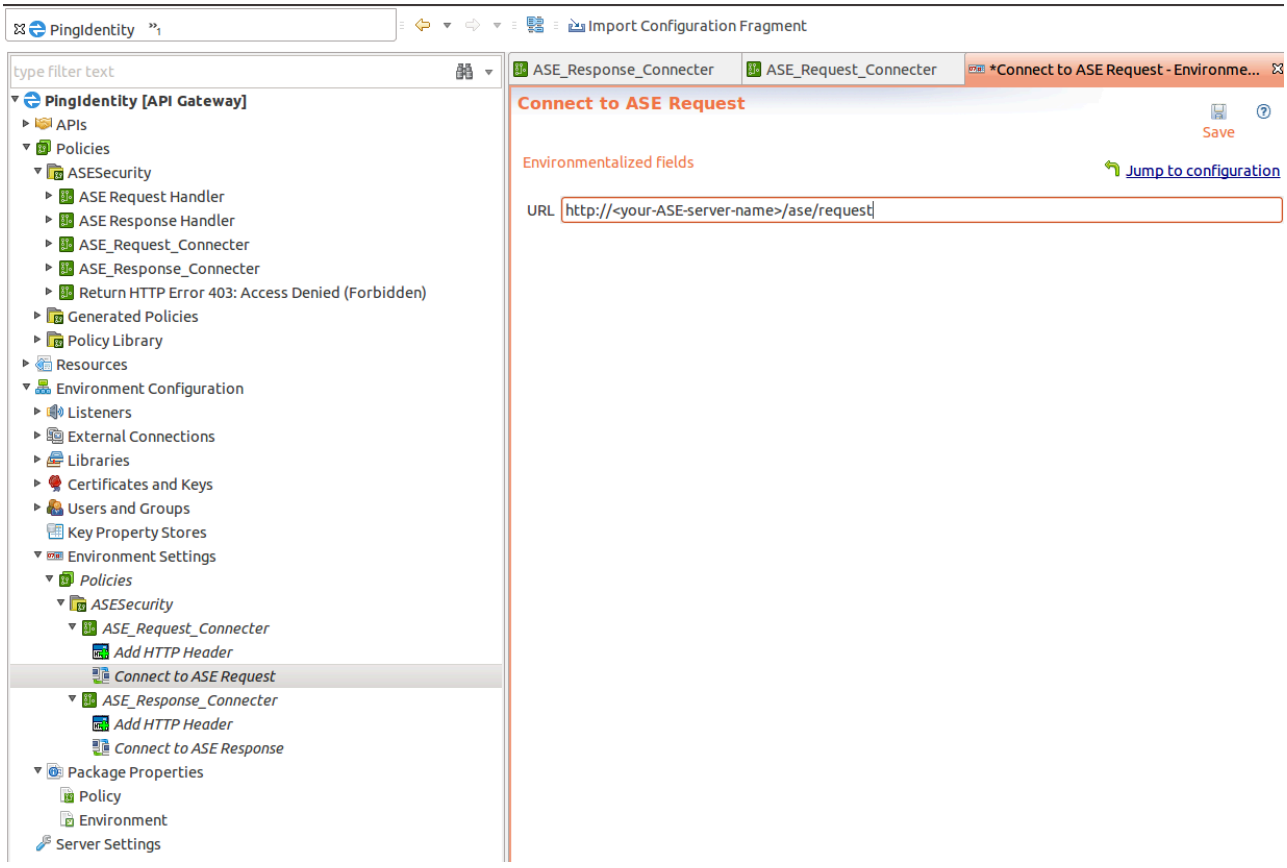
b. In the **Select OAuth Cache** pop-up window, select the OAuth token store.



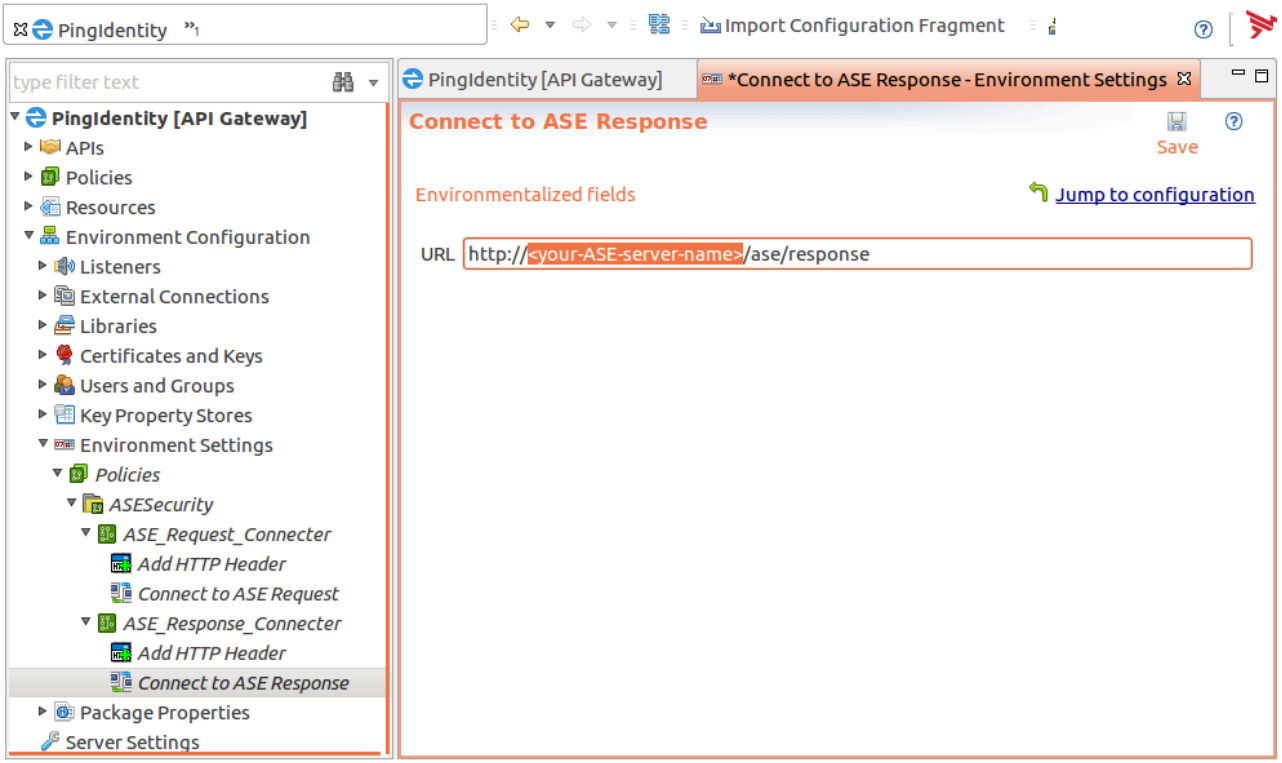
6. After the Axway Policy is imported, click **Environment Settings** in the left-hand column and Click **Add HTTP Header**. In the HTTP Header Value field, enter the ASE authentication token that was created.



7. After the Axway Policy is imported, click **Environment Settings** in the left-hand column and click **Connect to ASE Request** under **ASE_Request_Connector**. Enter the IP address or the hostname of your ASE in the **URL** field as shown in the screen shot:

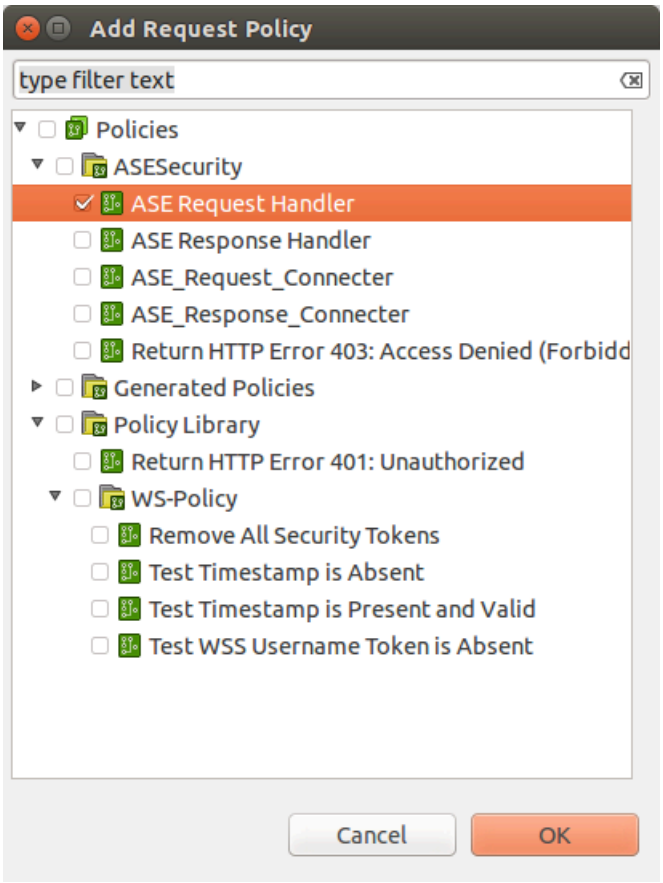


8. In the **Environment Settings** in the left-hand column, click **Connect to ASE Response** under **ASE_Response_Connector.** Enter the IP address or the hostname of your ASE in the **URL** field as shown in the screen shot:

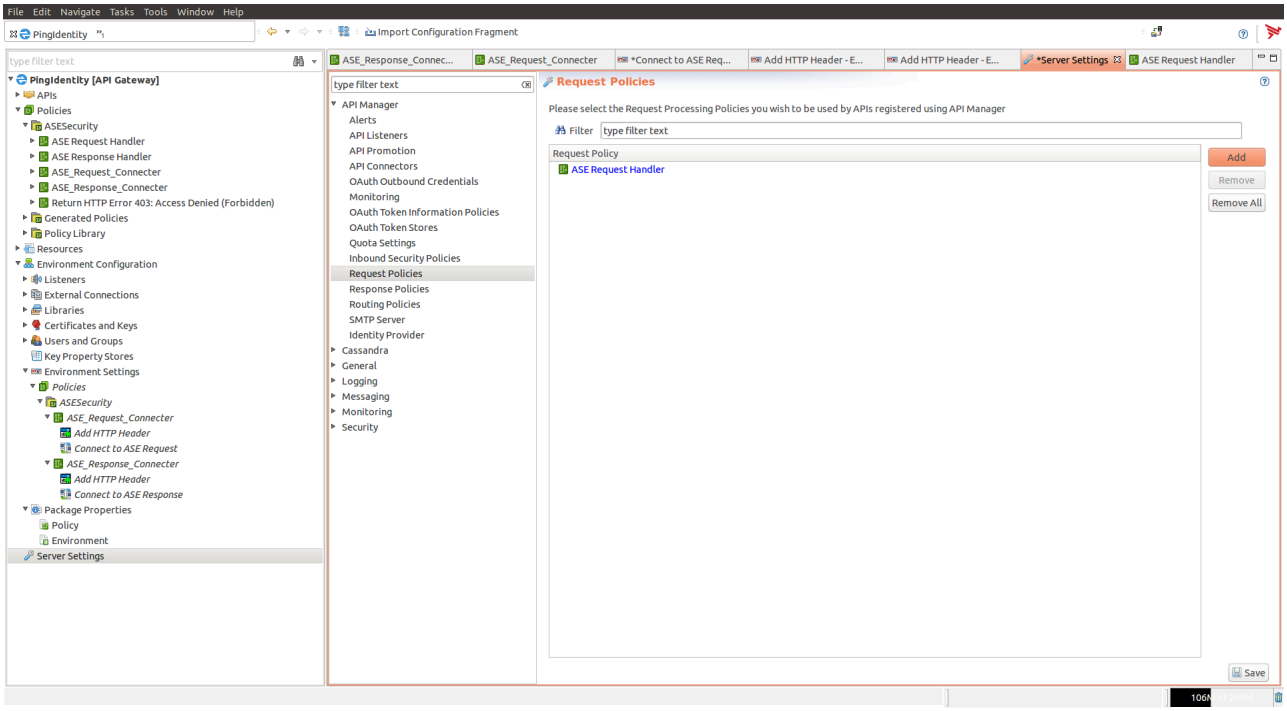


9. In the left pane of the window, click **Server Settings**.
10. In the Server Settings window, double-click **Request Policies** under **API Manager**

11. In the **Add Request Policy** pop-up window, check the **ASE Request Handler** and click **OK**



12. Click **Add** and then **Save**



13. Repeat step 9-10 for Response Policies.

14. Deploy the Policies by clicking **Deploy**.

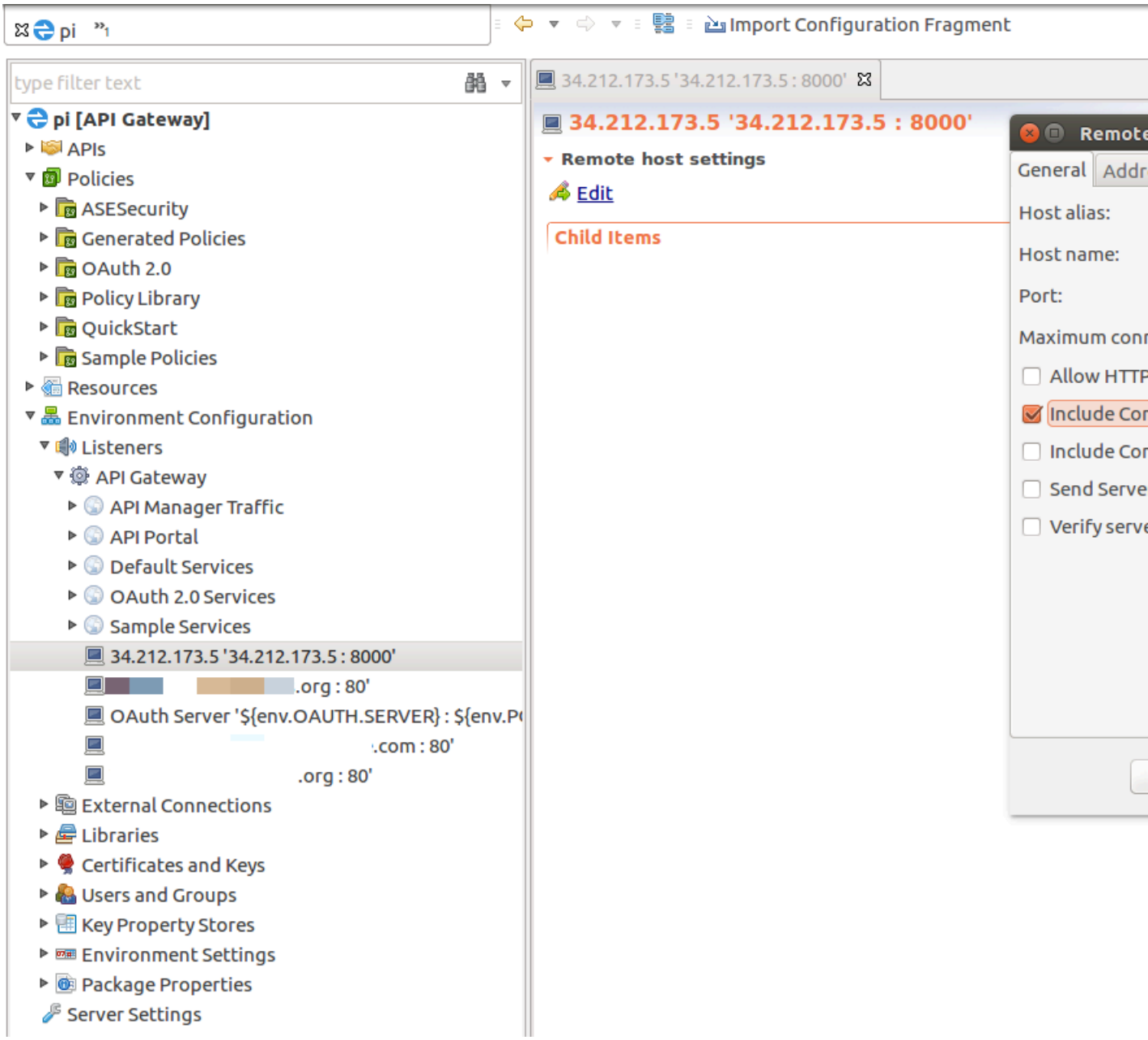
Configure ASE persistent connection

You can optionally configure TCP keep-alive connections in the `ase.conf` file of ASE. Following is a snippet of `ase.conf` displaying the `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 by completing the following steps:

1. Click on **Environment Configuration**
2. Under **Environment Configuration**, click **Listeners** > **API Gateway**.
3. Click On your ASE IP address in **Sample Services**
4. In the **Remote Host Settings** pop-up window, un-check **Allow HTTP 1.1**
5. Check **Include Content Length in request**. Make sure all other options are not selected.
6. Click **OK** and Deploy the policy

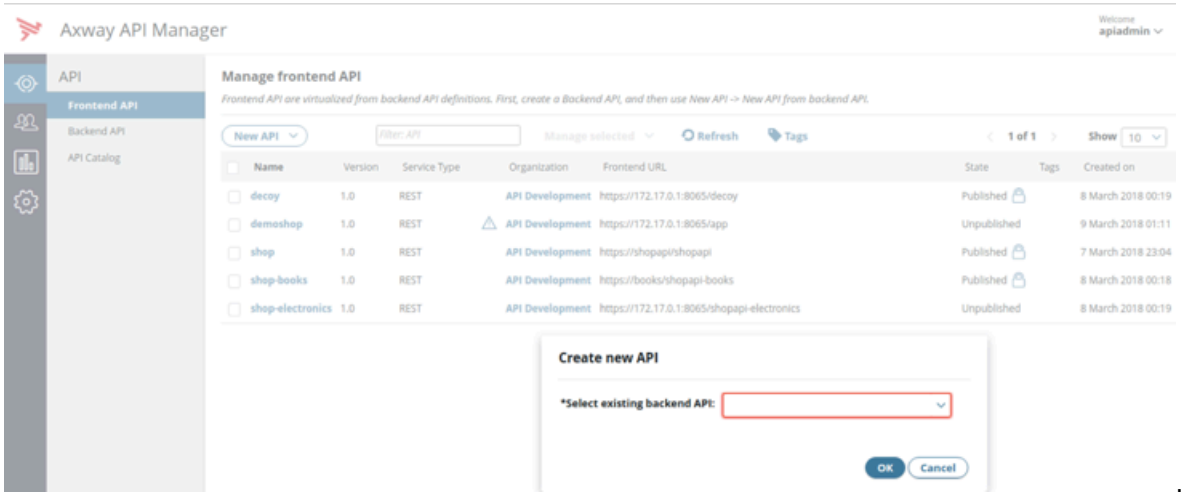


Axway API Manager configuration

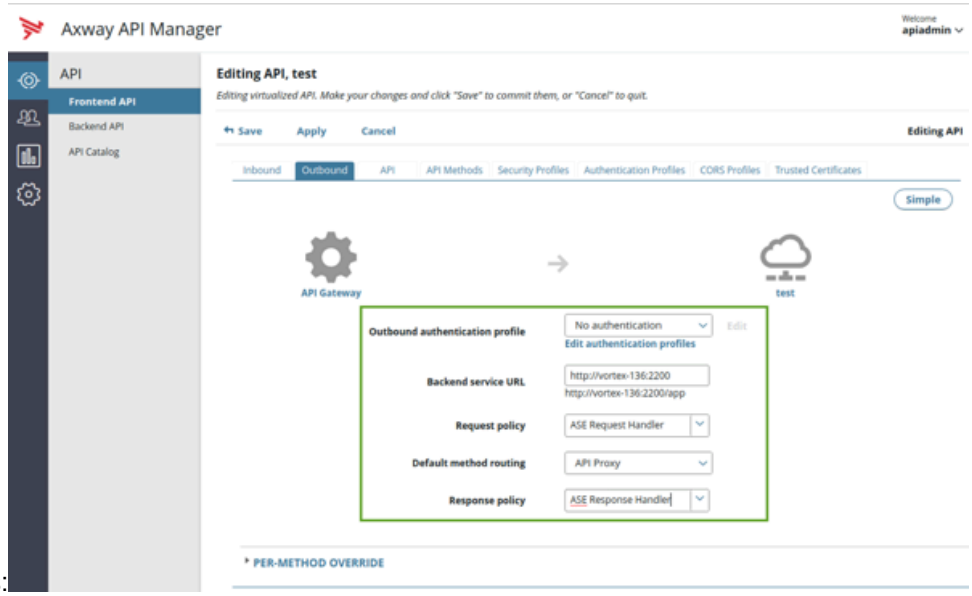
Complete the following steps to configure Axway API Manager:

1. Login to the Axway API Manager.

2. In the Axway API Manager, click **Frontend API** and **Create new API**



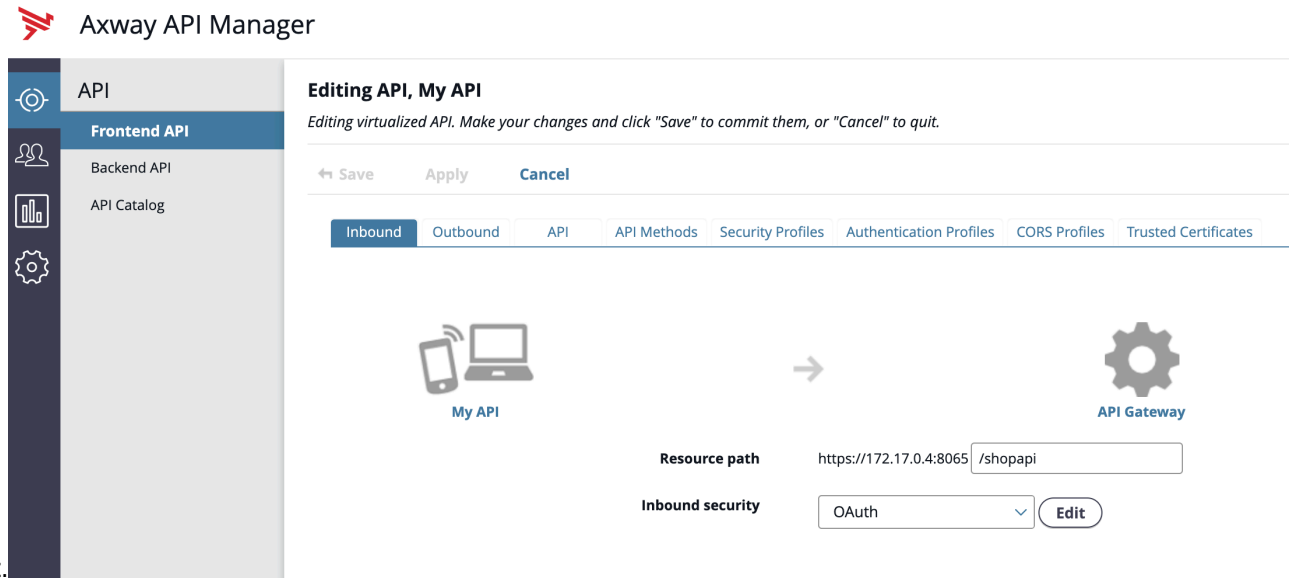
3. Click **Outbound tab** and enter Backend Service URL (your backend application server) and Request Policy details:



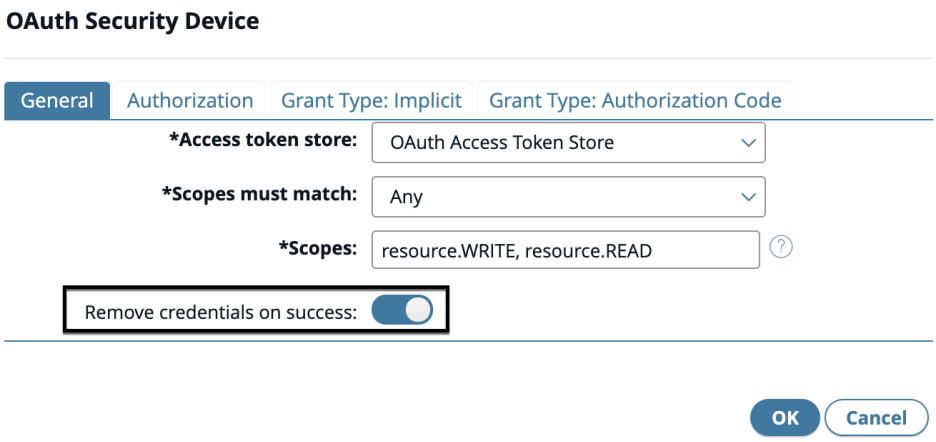
Configuration for capturing OAuth: To capture OAuth token based attacks, complete the following steps:

1. In the API Manager, click on **Frontend API** > **Inbound** tab.

2. From the **Inbound security** drop-down list, select **OAuth** and click **Edit**.



3. In the **OAuth Security Device** window, disable **Remove credentials on success** radio button.



API discovery

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 JSON is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

Axway API Manager configuration for PingIntelligence Dashboard

The PingIntelligence Dashboard pulls the API definition from Axway API Manager and converts them to a JSON format compatible with ASE. The Dashboard needs certain tags to be configured in Axway API Manager for it to import the normal and decoy API definitions. The following topics provide more information on configuring tags in Axway API Manager and configuring tags for the decoy API:

- *Configure tags in API Manager*
- *Configure tags for decoy API*

Configure tags in API Manager

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 URL parameters to be captured by PingIntelligence Dashboard for adding to ASE API JSON definition.

Tagging the API for AI processing:

You need to configure `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 5 APIs traffic to be processed using the AI engine, then apply the `ping_ai` tag on those 5 APIs.

In the Axway API Manager, click on **Frontend API > API** tab. In the API tab, navigate to 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 the `enable_blocking` in 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`.

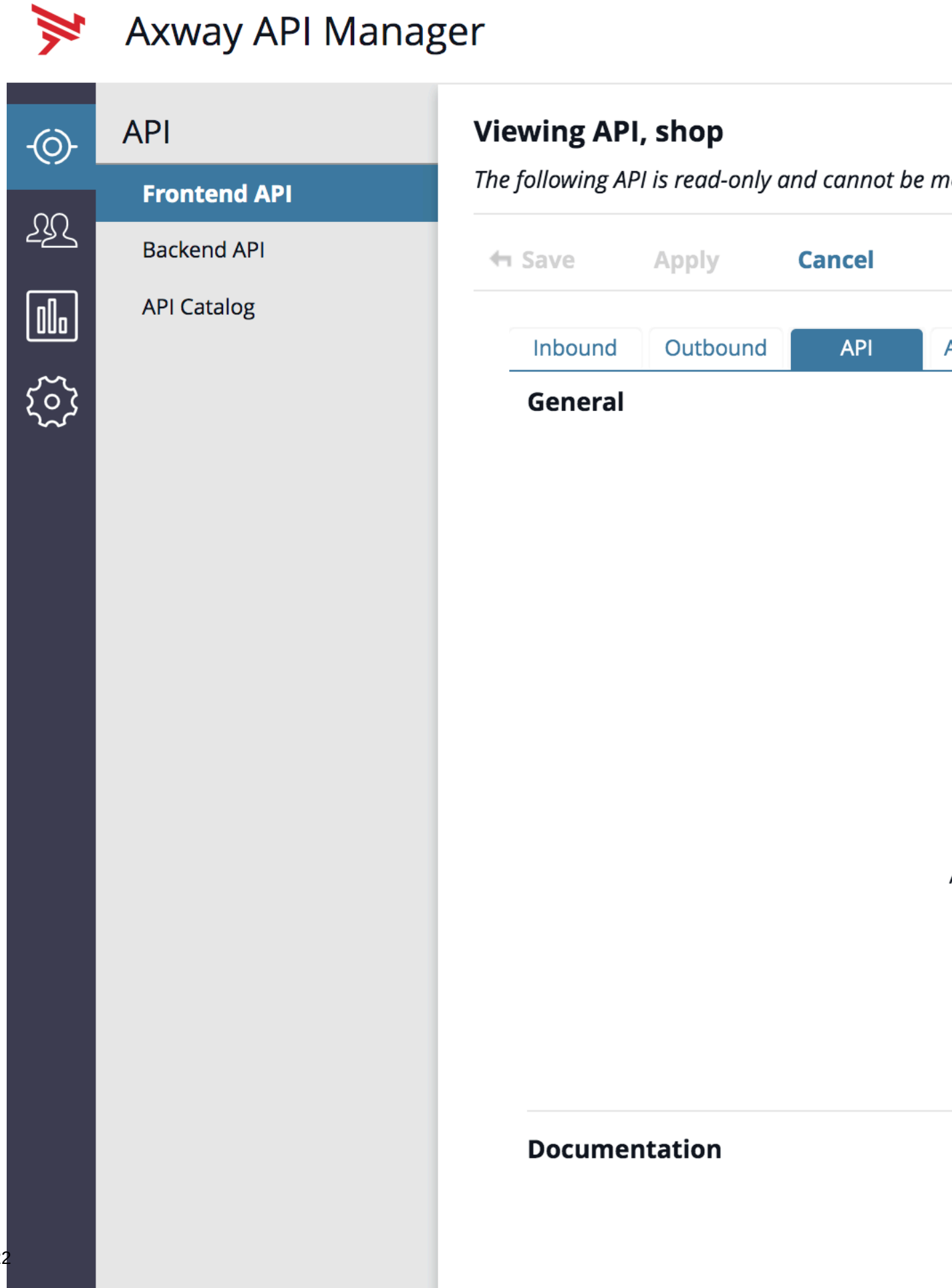
Tags for Cookie and Login URL (Optional)

If your APIs use a cookie or log in URL then configure the following two tags and values for a cookie and login URL. In the Axway API Manager, click **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.

The following illustration shows the tags to be added:

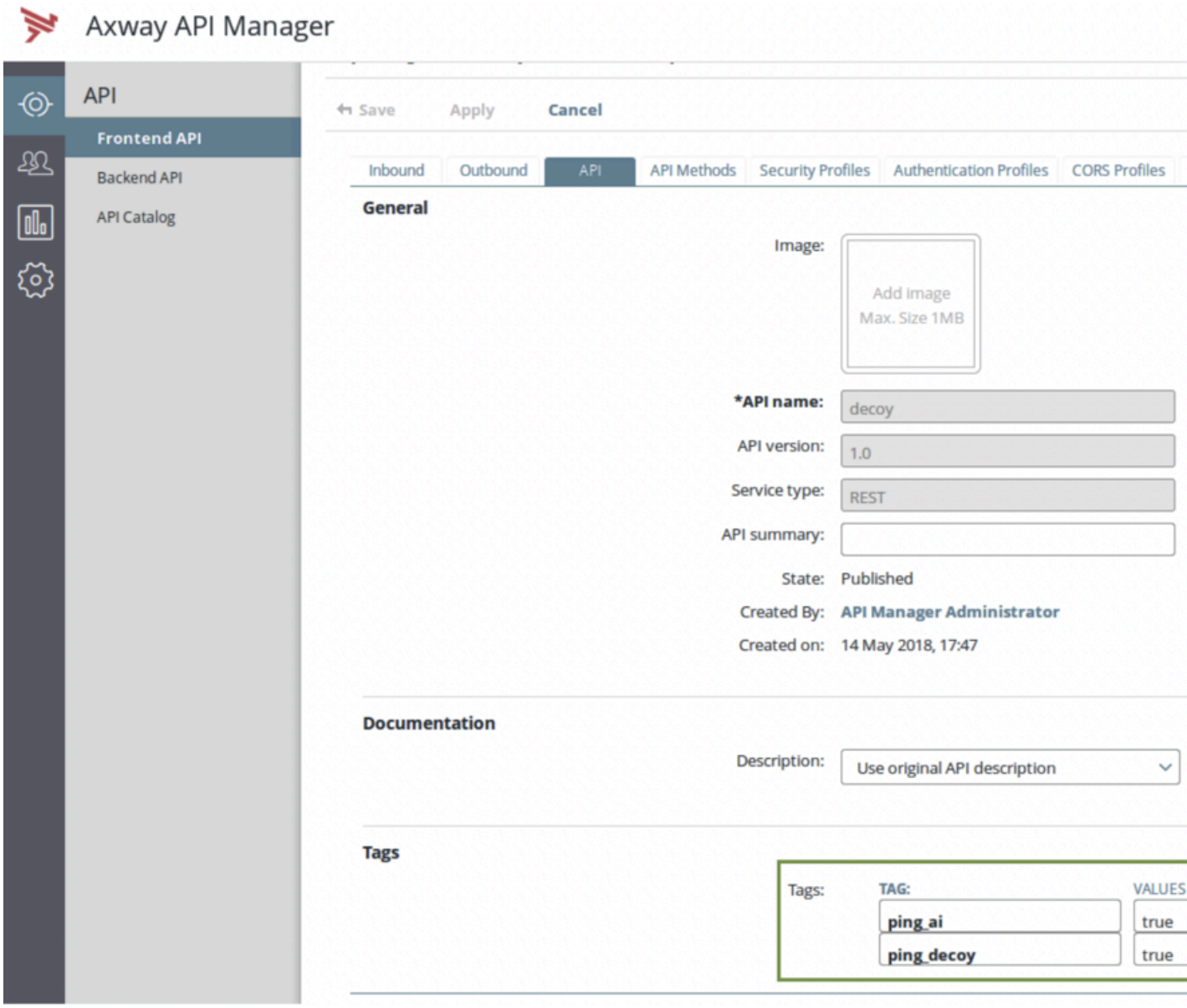


Configure tags for decoy API

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 Decoy API configured in Axway API Gateway is redirected to ASE which functions as the backend server. ASE sends a preconfigured response, like `200 OK`, for requests sent to a Decoy API.

You need to configure the following **TAGS** and **VALUES** in the **API** tab for **Frontend API** in Axway API Manager:

- `ping_ai - true`
- `ping_decoy - true`



API JSON for decoy API: The converted API JSON will have the decoy section configured as highlighted in the following JSON file:

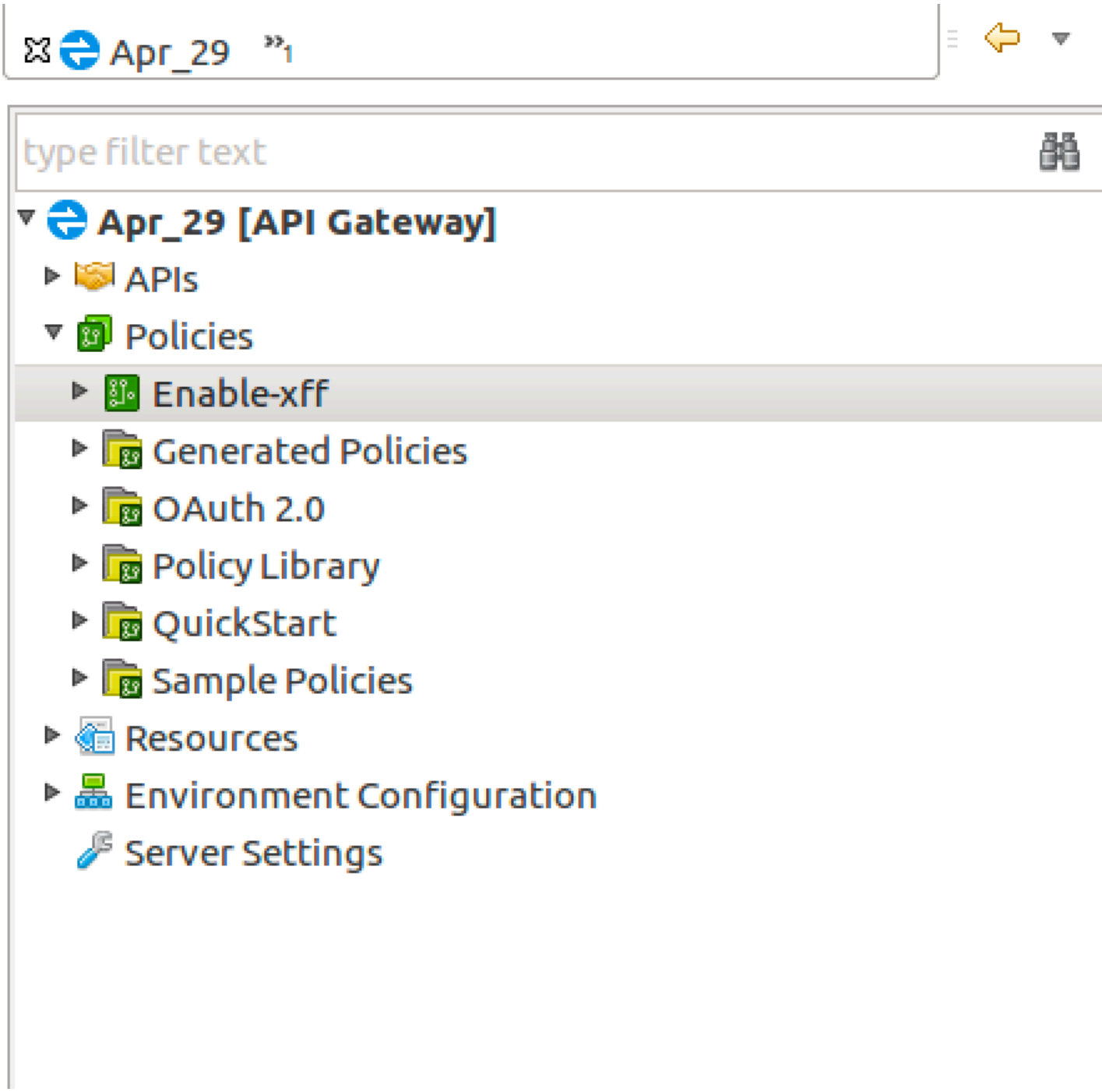
```
{
```

```
    "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": []
        }
    }
}
```

Axway XFF policy for decoy APIs

PingIntelligence provides an XFF policy for your decoy APIs. 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.

Follow the steps 1-4 of *Axway Policy Studio configuration* to import the XFF policy. Deploy the XFF policy after importing.



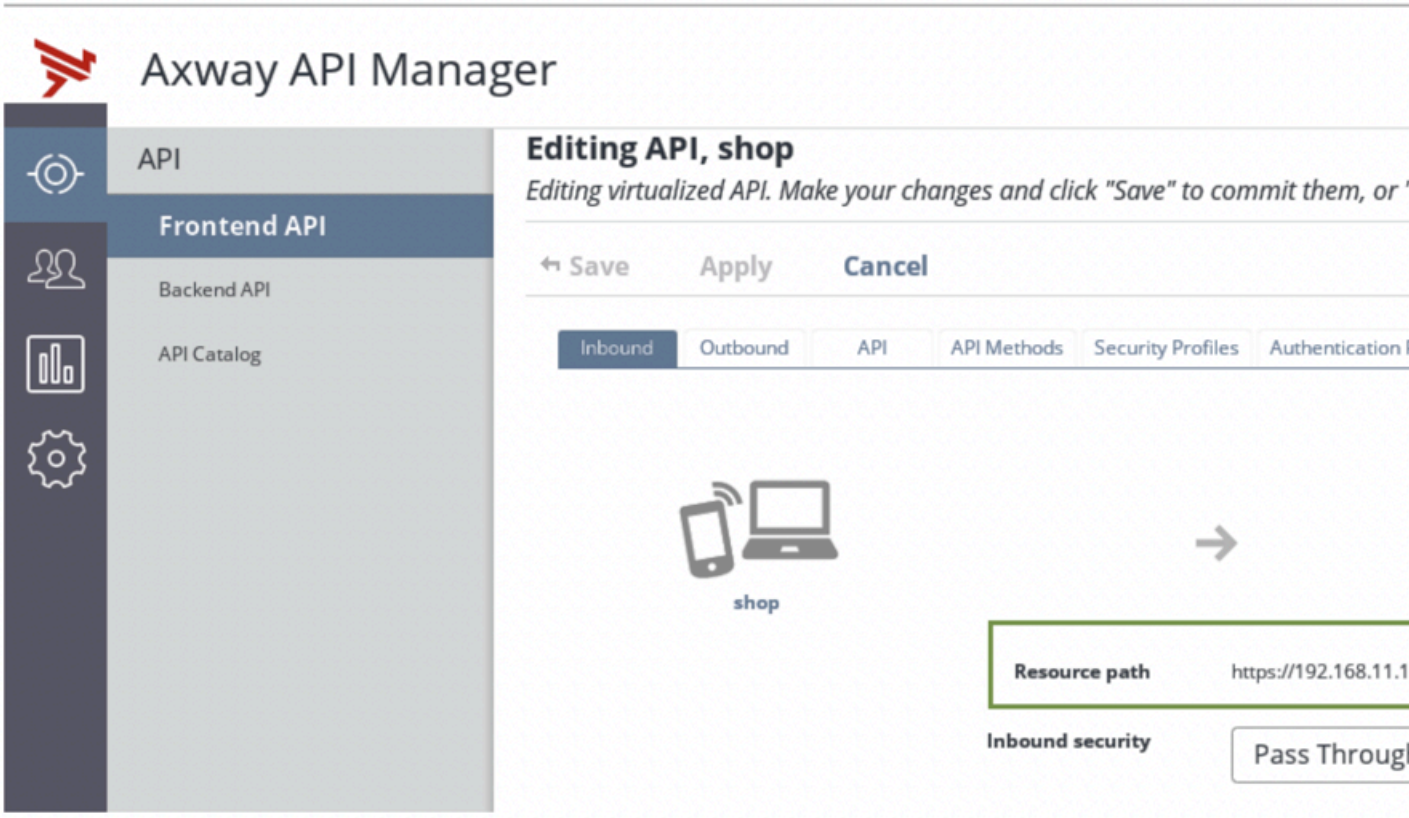
OAuth2 Token and API Keys

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 ASE API JSON. PingIntelligence's AI engine considers API Key values only in request headers or the query string.

Similarly, if you have configured 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 OAuth2 token.

Following API JSON file shows the converted parameters. The `protocol`, `url`, and `hostname` are picked from the values that you configure in **Resource path** when you create the Frontend API.




```
{
    "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 API gateway integration

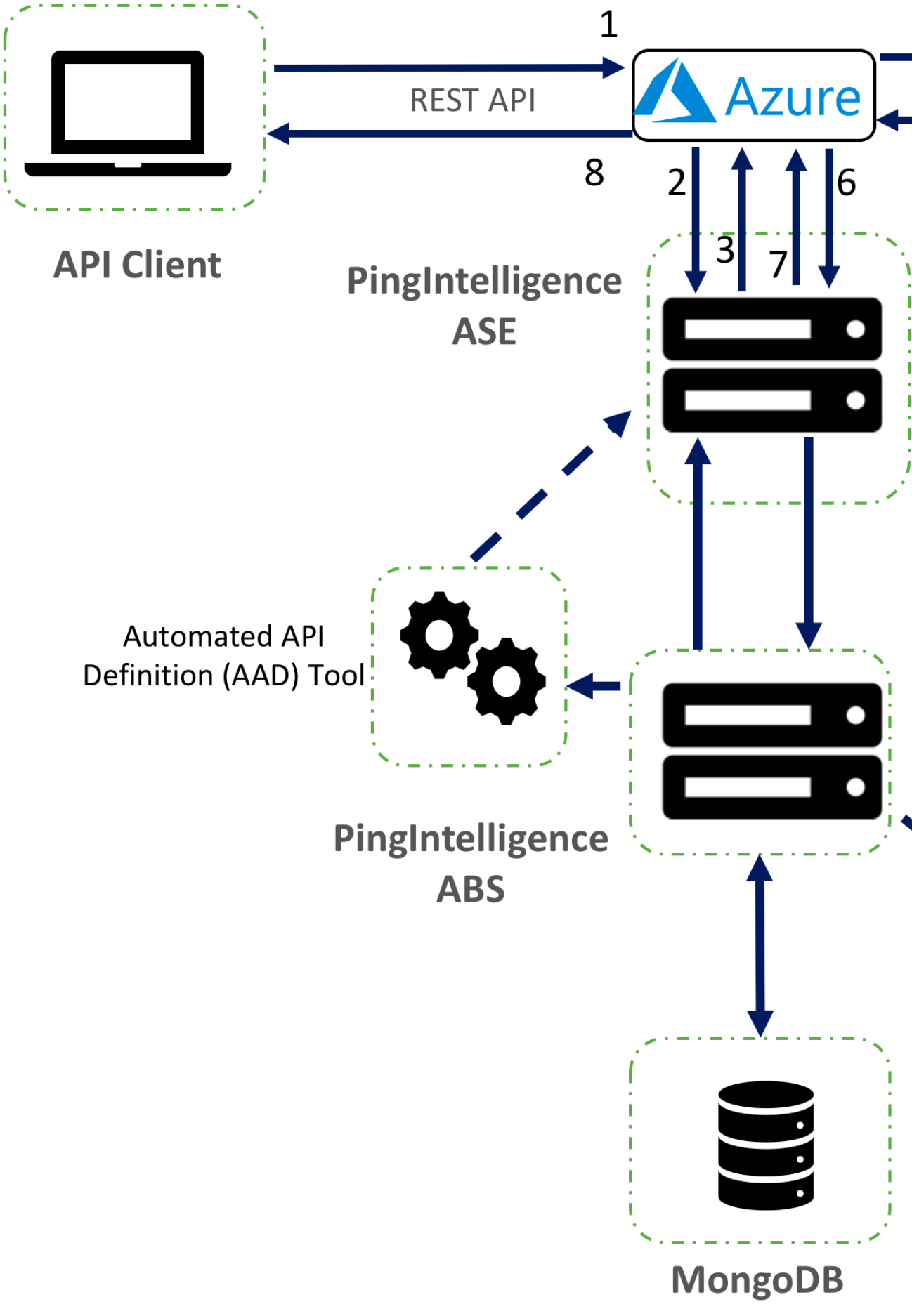
Azure APIM sideband integration

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. PingIntelligence policy for Azure also supports detecting attacks based on the username.

The APIM PingIntelligence policy works in the following two configurable mode:

- **Asynchronous mode**: When the PingIntelligence policy is configured in the Asynchronous mode, APIM does not wait for a response from PingIntelligence 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



Here 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 IP, cookie, OAuth2 token or API key on the PingIntelligence AI engine generated Blacklist. 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.
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.

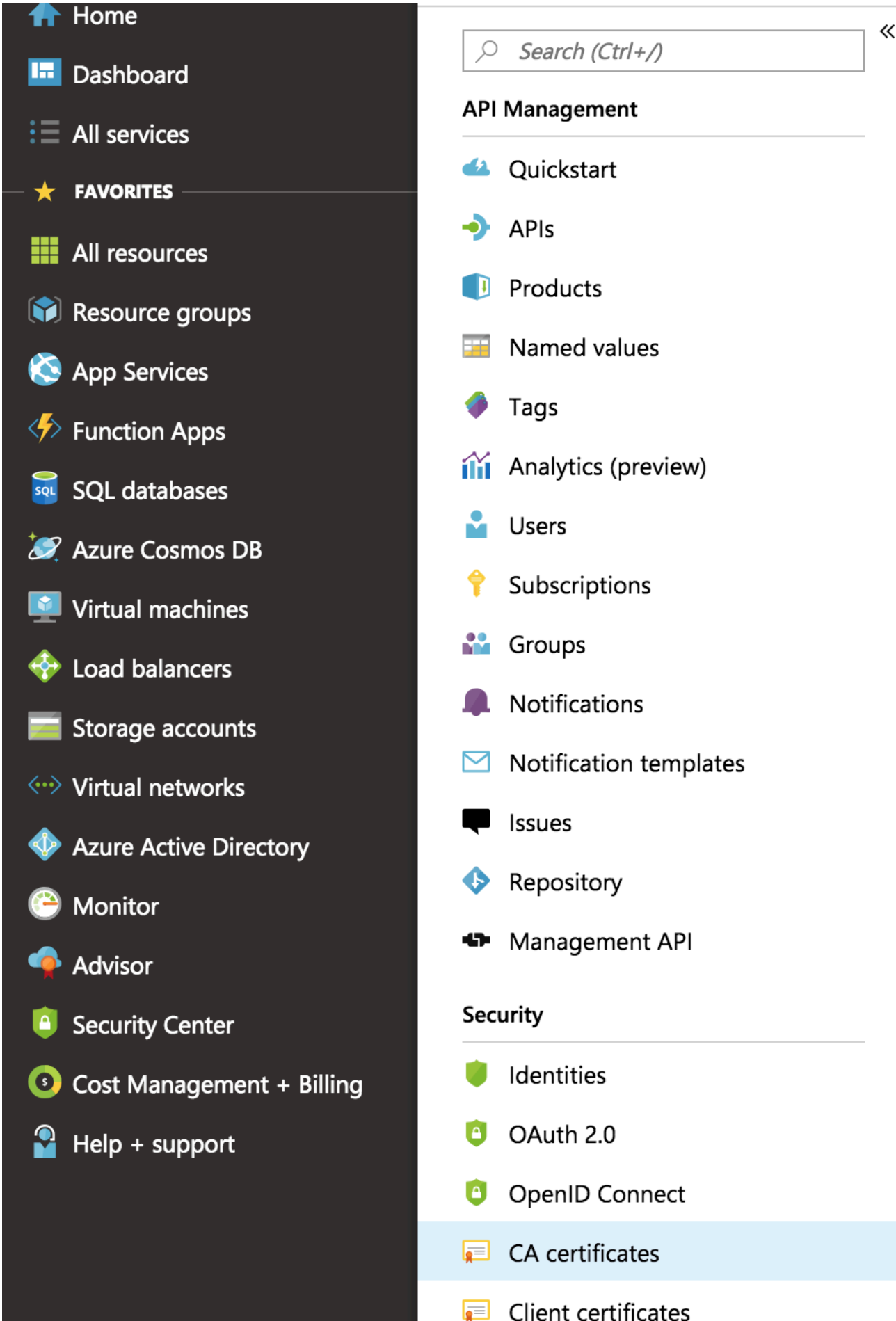
Prerequisites

Complete the following prerequisites before deploying the PingIntelligence policy on APIM:

Prerequisite:

- Confirm that the Azure API Management Service is available
- **Version** : The PingIntelligence policy supports Azure APIM Q2CY2020 version. If you are using any other version, contact Ping Identity support.
- Confirm that the APIs to which you want to apply the PingIntelligence policy are available

- **Configure CA certificate in APIM**: If you want to use the ASE self-signed certificate, then configure the CA certificate from the **Security -> CA certificates**



- **PingIntelligence policy application**

  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
  - For individual APIs
  - For a specific operation in the API
- **PingIntelligence software installation**

  Install and configure PingIntelligence software. Refer to the PingIntelligence deployment guide for your environment.
- **Verify that ASE is in sideband mode**

  Check that 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                     : 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)
```

  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.
- **Enable sideband authentication:** 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
```

- **Generate sideband authentication token**

  A token is required for APIM to authenticate with ASE. To generate the token in ASE, enter the following ASE command:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use.

Deploy PingIntelligence policy

PingIntelligence provides an XML policy file to integrate PingIntelligence and Azure API Management Service. 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 PingIntelligence AI engine for analysis.

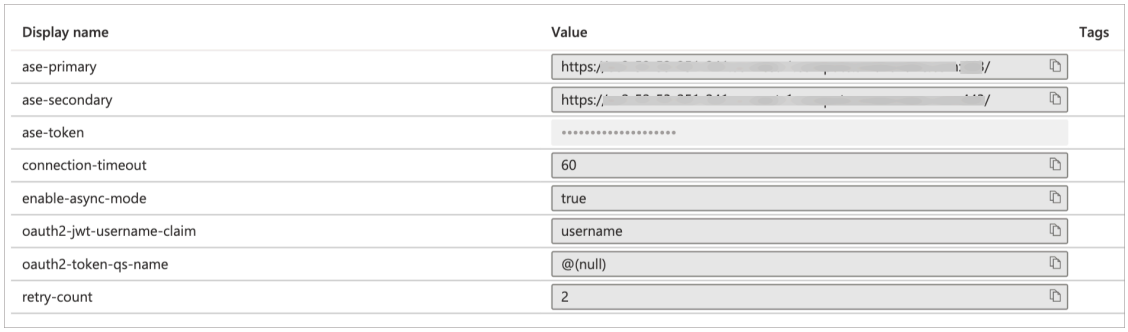
Complete the following steps to deploy the PingIntelligence policy:

1. *Download* the PingIntelligence policy XML file from the Sideband Integration section of the download page
2. Login to your Azure account and create the following **Named value** in your API Management service
   - **ase-primary**: The primary ASE node.
   - **ase-secondary**: The secondary ASE node. The traffic is redirected to the secondary ASE node if the primary ASE node is not reachable.

     ⓘ **Note:** Make sure that the ASE primary and secondary IP address is followed by a `/`.
   - **ase-token**: The authentication token for secure communication between Azure API Management service and ASE.
   - **connection-timeout**: The number of seconds for which the API Management Service waits for ASE to respond.
   - **enable-async-mode**: Set the value to **true** to enable asynchronous mode between APIM and ASE. When the asynchronous mode is enabled, the Azure gateway does not wait for a response from ASE and sends the request to the backend server. The ASE performs detailed API activity reporting and attack detection without blocking of attacks. If you do not want to enable asynchronous mode, set the value to **false**. In this case, the Azure gateway does not send the API request to the backend server, until it receives a response from ASE.
   - **oauth2-jwt-username-claim**: JWT claim name for username.
   - **oauth2-token-qs-name**: The name of the query string parameter that contains the OAuth token. If you choose not to intercept the OAuth tokens coming as part of query string, then set the value to `@(null)`.

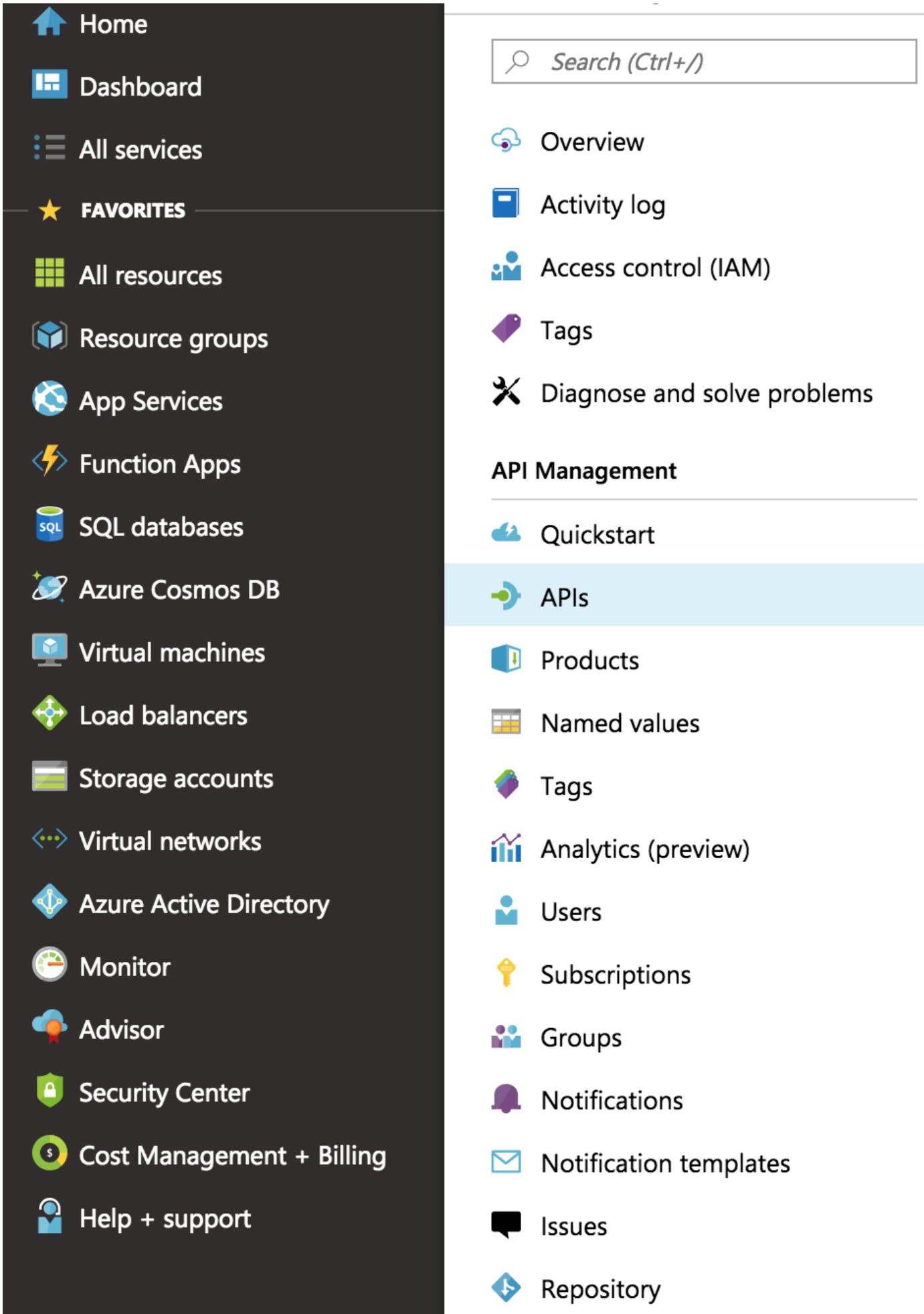
     ⓘ **Note:** The PingIntelligence policy extracts the OAuth token from the query string, configured in **oauth2-token-qs-name**. A new Authorization header- `Authorization: Bearer <OAuth token>` is added to the metadata sent to ASE. If there is an existing Authorization header, the token is prepended 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.
   - **retry-count**: The number of times APIM tries to connect to ASE.

   If you change any of the **Named Values** after the policy is operational, it takes 60-seconds for the change to be applicable. For example, if you change the ase-primary node IP address, the new IP address would take effect only after 60-seconds.

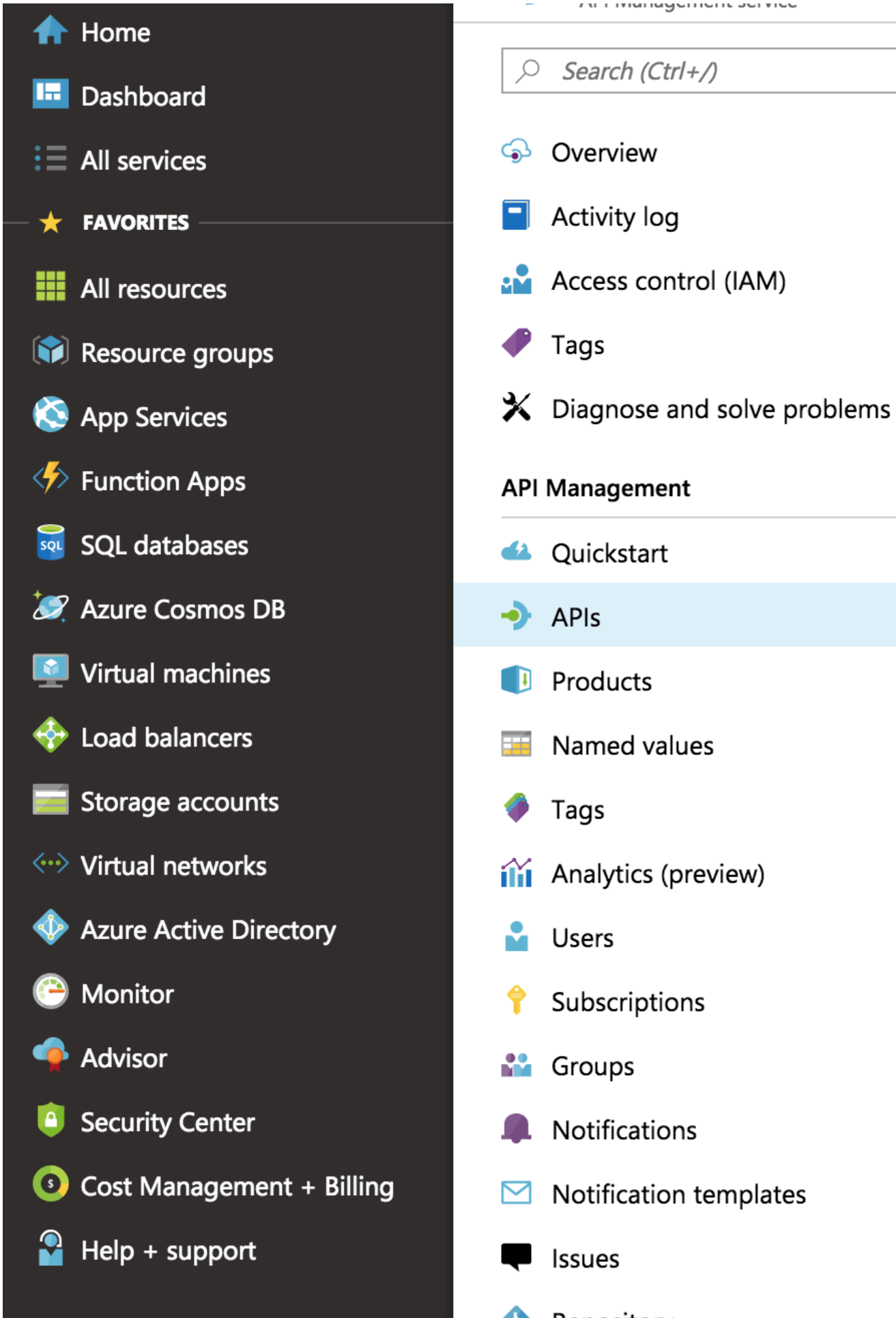


3. Open the downloaded PingIntelligence policy XML file and copy the policy at the desired level: **All APIs**, individual APIs, operation level, or Group of APIs. Click on **Policies** in the **Inbound processing** UI box and paste the policy.

ⓘ **Note:** The PingIntelligence policy does not validate the authenticity of a JWT. Configure the PingIntelligence policy after <validate-jwt> policy.



4. Click on the **Save** button to save the



> ⓘ **Attention:** 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.

API discovery

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 JSON is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

Integrate PingIntelligence

After the policy deployment is complete, refer to the following topics for next steps:

It is recommended to read the following topics (part of the admin guides) apart from reading the *ASE* and *ABS AI Engine* Admin Guides:

- *Customizing ASE ports* on page 115
- *API naming guidelines* on page 153
- Adding APIs in *Sideband ASE* on page 144. You can add individual APIs or you can configure a global API. For more information, see*API discovery and configuration* on page 321.
- *Configure ASE to ABS connectivity* on page 173

After you have added your APIs in ASE, the API model needs to be trained. The training of an API model is executed in the ABS AI engine. The following topics provide information on important topics, however it is a good practice to read the entire ABS Admin Guide.

- *AI Engine training* on page 312
- *API reports using Postman* on page 386
- *Access PingIntelligence Dashboard* on page 17

Configure ASE persistent connection

You can optionally configure TCP keep-alive connections in the `ase.conf` file of ASE. Following is a snippet of `ase.conf` displaying the `enable_sideband_keepalive` variable. The default value is set to `false`.

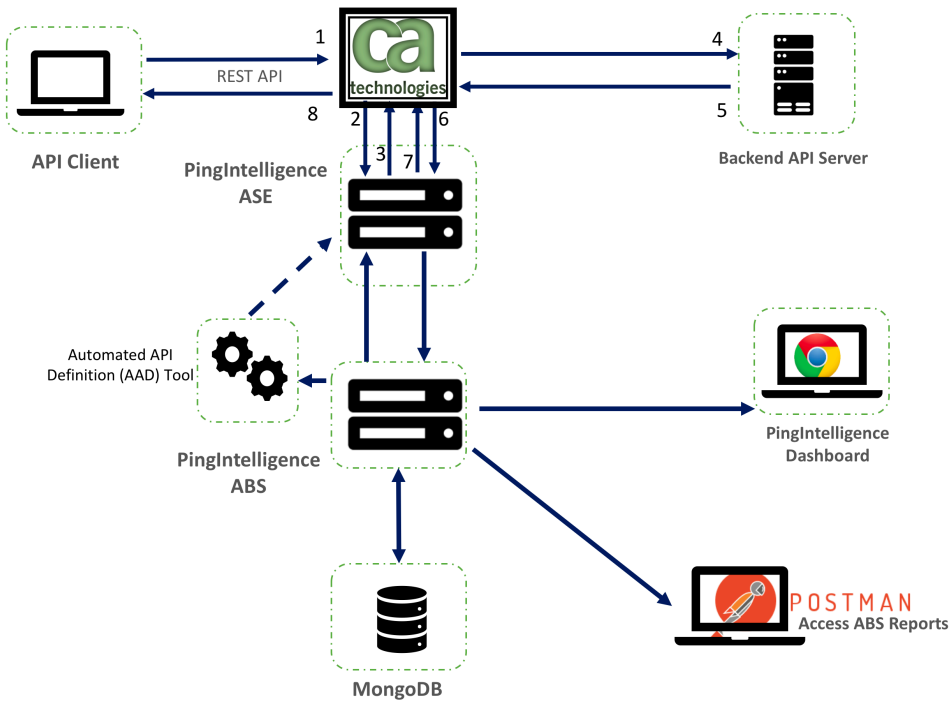
```
; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false
```

CA API gateway integration

PingIntelligence - 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:



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 blacklist 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. 8. CA sends the response received from the backend server to the client.

PingIntelligence encapsulated assertions include capabilities for enhanced sideband performance and availability including:

- **Persistent SSL sessions** - Support for flowing sideband calls across a persistent 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.

Prerequisite

Confirm that the following prerequisites are met before deploying the PingIntelligence integration.

Prerequisite:

- **CA API Gateway Policy Manager** - PingIntelligence was developed with and qualified with CA API Gateway 9.4 (contact PingIdentity for other supported releases). Use the included Policy Manager to configure the gateway..
- **PingIntelligence software installation**

  PingIntelligence 4.0 software is installed and configured. For installation of PingIntelligence software, refer to the *manual* or *automated* deployment guides.
- Java must be installed on the system from where the bundle is imported into the CA API gateway
- **Verify that ASE is in sideband mode** Confirm that ASE is operating in `sideband` mode by running the following command in the 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                     : 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.
- **Enable sideband authentication:** 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
```

- **Generate sideband authentication token**

  A token is required for CA to authenticate with ASE. This token is generated in ASE and configured in the policy XML file. To generate the token in ASE, enter the following command in the ASE command line:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use.

Install and configure the PingIntelligence bundle

Installing and configuring the PingIntelligence bundle for CA API gateway consists of following steps:

1. Configure properties in `pingintelligence-properties.bundle` file.
2. Import the bundle file and the properties file into the CA API gateway using the import script.
3. Configure a certificate and the ASE token using CA API Policy Manager

Configure PingIntelligence bundle

Complete the following steps to configure the CA API gateway PingIntelligence policy:

1. Download the PingIntelligence policy files from the *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
3. Edit the `pingintelligence-properties.bundle` to configure the following properties:
   - `ase_host_https` and `ase_host2_https`: Primary and secondary PingIntelligence ASE 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 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 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 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 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 default value in sample file.
   - `ase_path_response` and `ase_path2_response`: Use default value in sample file.

Following is a sample `pingintelligence-properties.bundle` file:

```
<?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>
                    <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>
```

Import PingIntelligence policy

After the PingIntelligence bundle is configured, import it into the CA API gateway. PingIntelligence provides a script to import the policy. Complete the following steps to import the bundle:

1. Open the `import_pingintelligence.sh` file in an editor.
2. Configure the following values:
   - `GW`: API gateway hostname and port
   - `GW_user admin:password`: API gateway username
   - `GW_PASS_B64`: A base64 encoded password used to encrypt/decrypt secure passwords
3. Run the `import_pingintelligence.sh` script. After the import script is run, the PingIntelligence policy is installed in the API gateway.

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.

Following is a sample `import_pingintelligence.sh` script:

```
!/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
```

Configure ASE token and certificate

After the bundle is imported into the CA API gateway, configure the certificate and ASE token using the CA API Policy Manager.

Configure the certificate: Complete the following steps to configure the certificate using CA API Policy Manager:

1. In CA API Policy Manager, navigate to **Tasks** > **Certificate, Keys and Secrets** > **Manage Certificates**
2. Click **Add** and complete the steps on the GUI to add a certificate.
3. In the **Specify Certificate Options** step (step 3 in GUI), select the **Outbound SSL Connections** checkbox and click **Next**.

4. In the **Configure Validation** step (step 4 in GUI), select the **Certificate is a Trust Anchor** checkbox and click **Finish**.

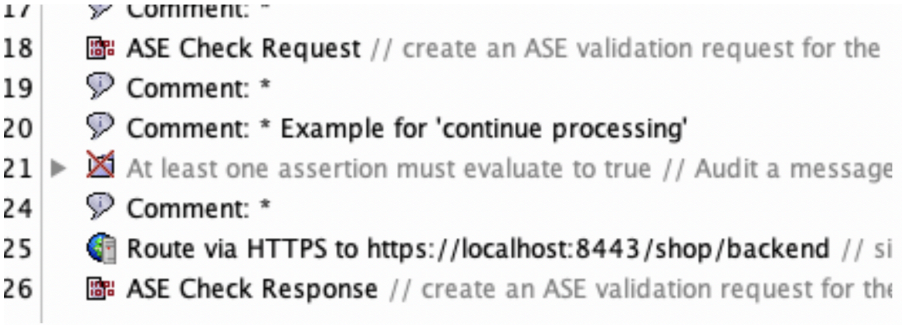
Configure ASE token: Complete the following steps in the CA API Policy Manager to configure the ASE token that was generated as part of *Prerequisite* on page 583.

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 on **Change Password**.
4. In the **Enter Password** pop-up window, enter the ASE token and click **Ok**.

Apply PingIntelligence policy

The bundle includes 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.

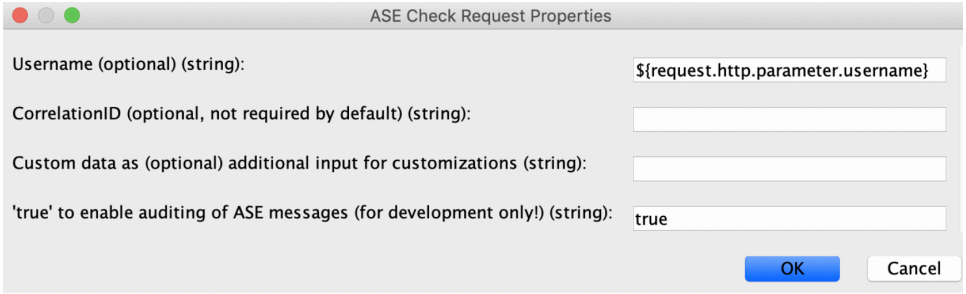
For service-level policies, each API will add two assertions, ASE Check Request and ASE Check Response. ASE Check Request is applied before routing the request to the backend. Whereas ASE Check Response is used after a call to the downstream endpoint (which is on line 25 in the screenshot below):



The ASE Check Request assertion is configured with the following:

ASE check request

ASE Check Request:



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 other if you are customizing to accommodate special use cases.

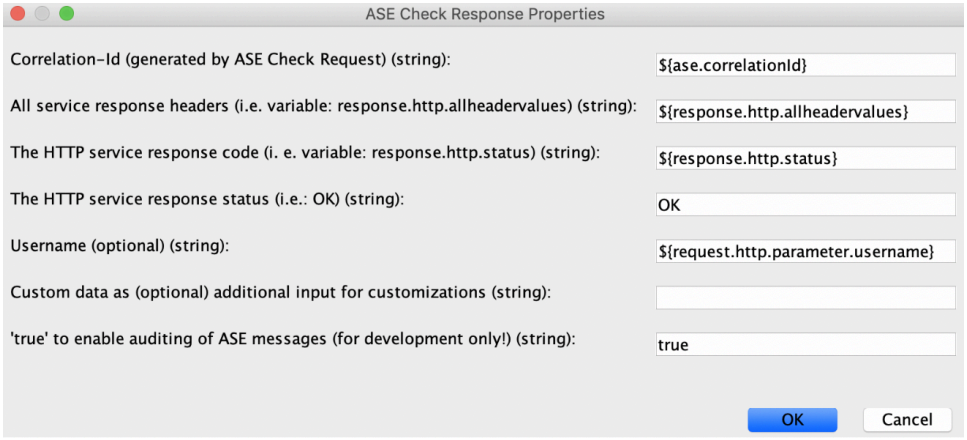
- **CorrelationID**: Optional – used if you want to override the `correlationId` which will otherwise automatically be assigned.
- **Custom data**: Optional - used to modify the internal of that assertion.
- **true**: Useful for users developing an API for debugging or auditing purposes.

The assertion has an output which 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:

- **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.



API discovery

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 JSON

is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

Integrate PingIntelligence

After the policy deployment is complete, refer to the following topics for next steps:

It is recommended to read the following topics (part of the admin guides) apart from reading the *ASE* and *ABS AI Engine* Admin Guides:

- *Customizing ASE ports* on page 115
- *API naming guidelines* on page 153
- Adding APIs in *Sideband ASE* on page 144. You can add individual APIs or you can configure a global API. For more information, see*API discovery and configuration* on page 321.
- *Configure ASE to ABS connectivity* on page 173

After you have added your APIs in ASE, the API model needs to be trained. The training of an API model is executed in the ABS AI engine. The following topics provide information on important topics, however it is a good practice to read the entire ABS Admin Guide.

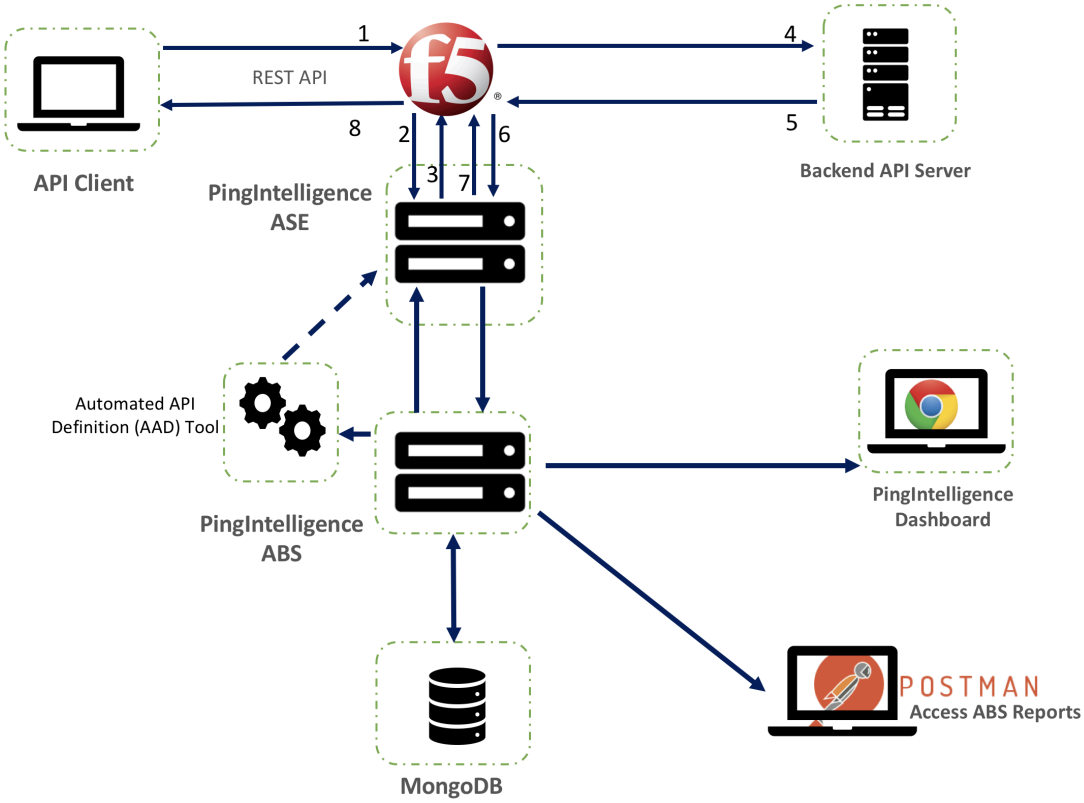
- *AI Engine training* on page 312
- *API reports using Postman* on page 386
- *Access PingIntelligence Dashboard* on page 17

F5 BIG-IP integration

F5 BIG-IP PingIntelligence 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).

This diagram depicts the architecture of PingIntelligence for APIs components along with F5 BIG-



IP:

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 token or API key in PingIntelligence AI engine generated Blacklist. 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.

Prerequisites

F5 BIG-IP and PingIntelligence sideband integration is qualified for 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.

F5 prerequisites:

- F5 BIG-IP with v13.1.0.8 software
- Knowledge of iRules LX in F5. Refer the F5 documentation for information on iRules.
- A Virtual Server is configured to front-end the incoming traffic. Make sure to apply HTTP profile to the virtual server.
- A valid F5 BIG-IP license and iRules LX is enabled in your setup.

PingIntelligence prerequisites:

This section assumes that you have installed and configured PingIntelligence software. For more information on PingIntelligence installation, see *PingIntelligence for APIs setup* on page 42 or *PingIntelligence manual deployment* on page 75

- Download the PingIntelligence policy from the *download* site.
- **Verify that ASE is in sideband mode:** Log in 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
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.
- **Enable sideband authentication:** For secure communication between F5 BIG-IP and ASE, enable sideband authentication by entering the following ASE command:

```
# ./bin/cli.sh enable_sideband_authentication -u admin -p admin
```

- **Generate sideband authentication token**

  A token is required for BIG-IP to authenticate with ASE. To generate the token in ASE, enter the following command in the ASE command line:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use in *Import and configure PingIntelligence policy* on page 593

Deploy PingIntelligence policy

Deploying PingIntelligence policy for F5 BIG-IP consists of the following steps:

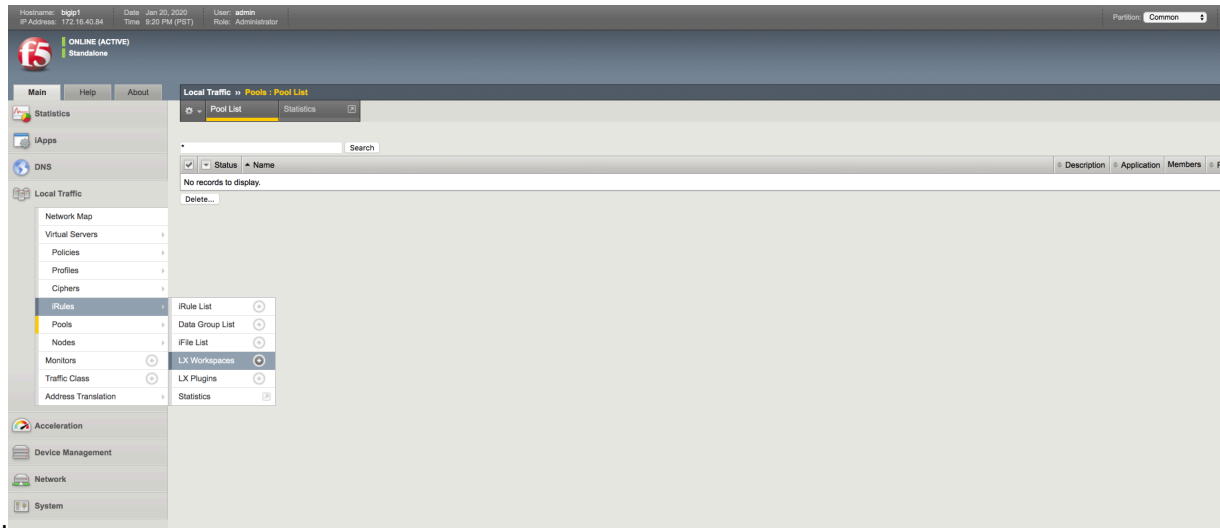
1. Import and configure PingIntelligence policy
2. Create an LX plugin
3. (Optional step) Add server pools for the backend and add virtual server for the frontend. If you already have frontend virtual servers and backend server pool, skip to next step.
4. Add iRule to the virtual server

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 Nodejs editor.

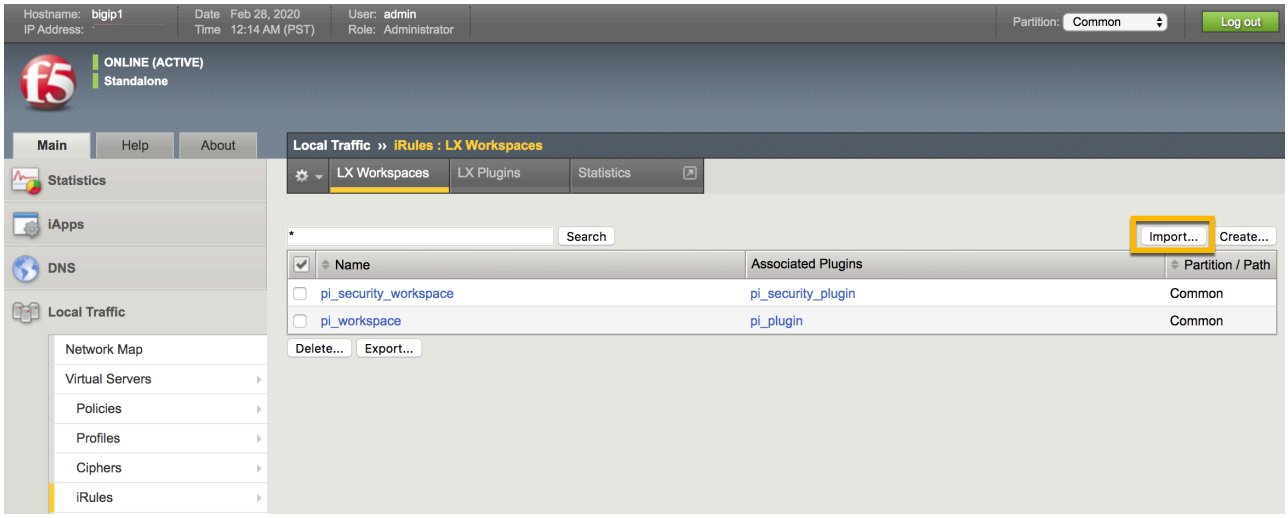
Import and configure PingIntelligence policy

Complete the following steps to import PingIntelligence policy in F5:

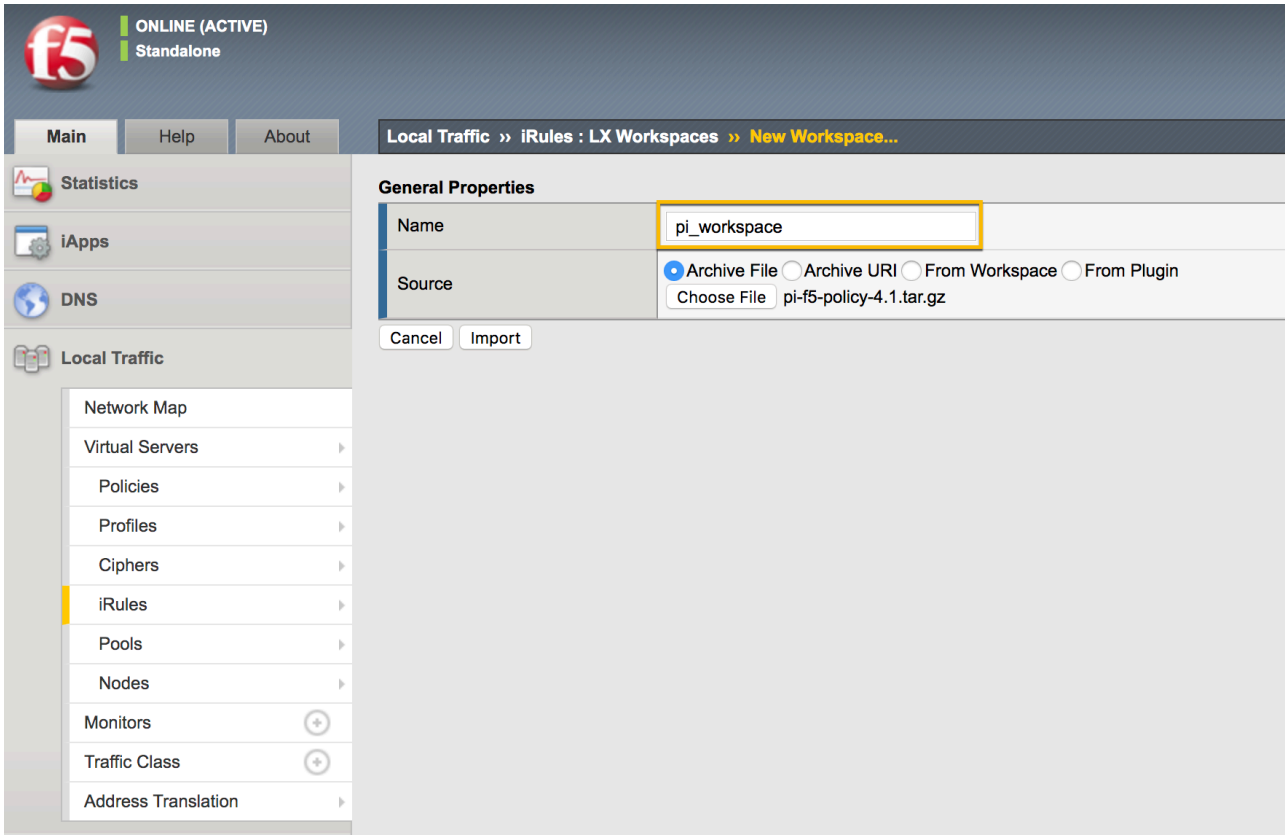
1. Login to your F5 UI and navigate to **Local Traffic** > **iRules** > **LX Workspaces**.



In the **Workspaces** tab, click on **import**.

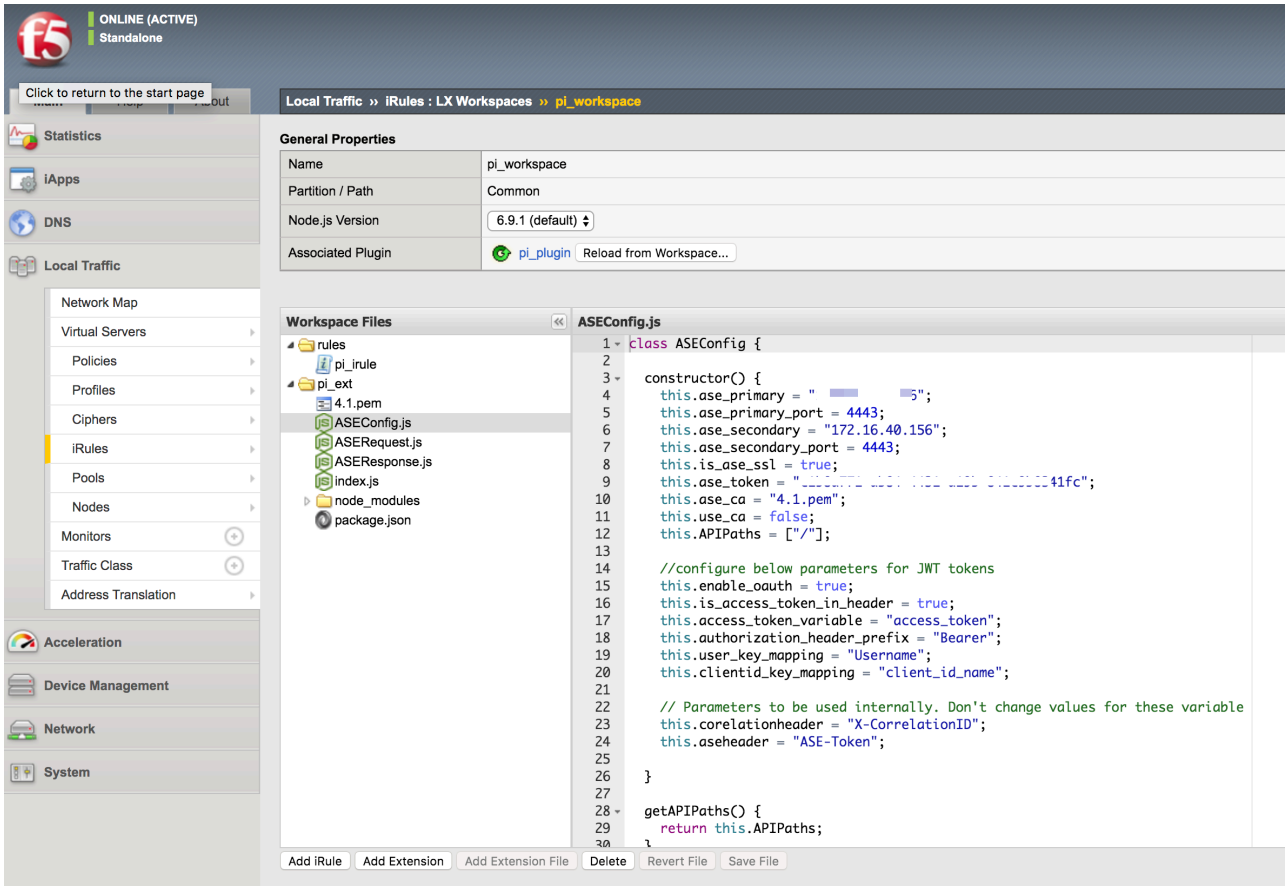


2. A Workspace import page is displayed. Enter the **Name** and choose the PingIntelligence policy that you downloaded from Ping Identity download site. Click on **Import**.



3. Clicking on **Import** creates an LX Workspace

4. Open the Workspace by clicking on it. The policy is pre-loaded with extension named `pi_ext`. Edit the ASE configuration by clicking on **ASEConfig.js** file. It opens the PingIntelligence policy in the editor:



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
`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
`APIPaths`	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.

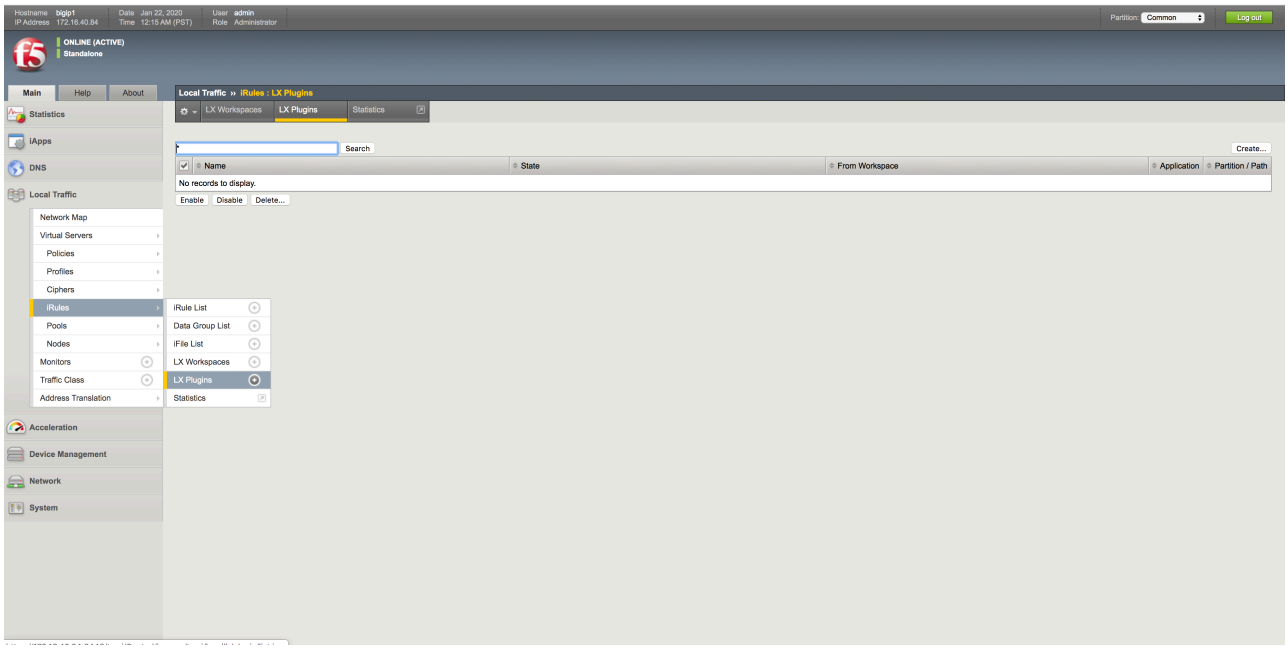
Variable	Description
`access_token_variable`	If the access token is present in `querystring`, then specify the key used for token.
`authorization_header_prefix`	If the access token is present in `authorization` header, then specify the prefix used for access token.
`user_key_mapping`	The location of `username` in JSON payload of JWT access token.
`clientid_key_mapping`	The location of client ID in JSON payload of JWT access token

Create LX plugin

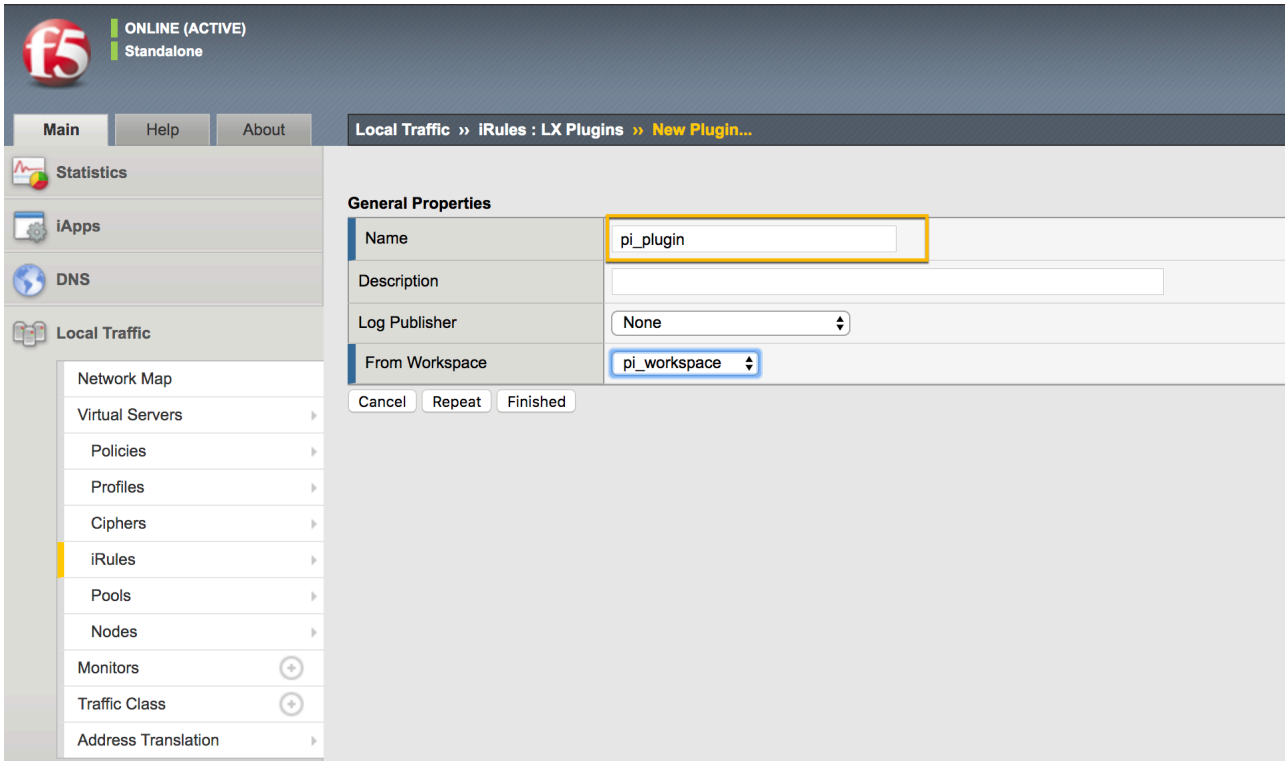
After importing and configuring the PingIntelligence policy, create an LX plugin with name **pi_plugin**.

Complete the following steps to create an LX plugin:

1. Navigate to **Local Traffic** > **iRules** > **LX Plugins**:



2. In the New Plugin page, click on **Create** to create a new plugin with name **pi_plugin**. Select the workspace that you created earlier from the **From Workspace** drop-down list and click on **Finished**.



(Optional) Create backend server pool and frontend virtual server

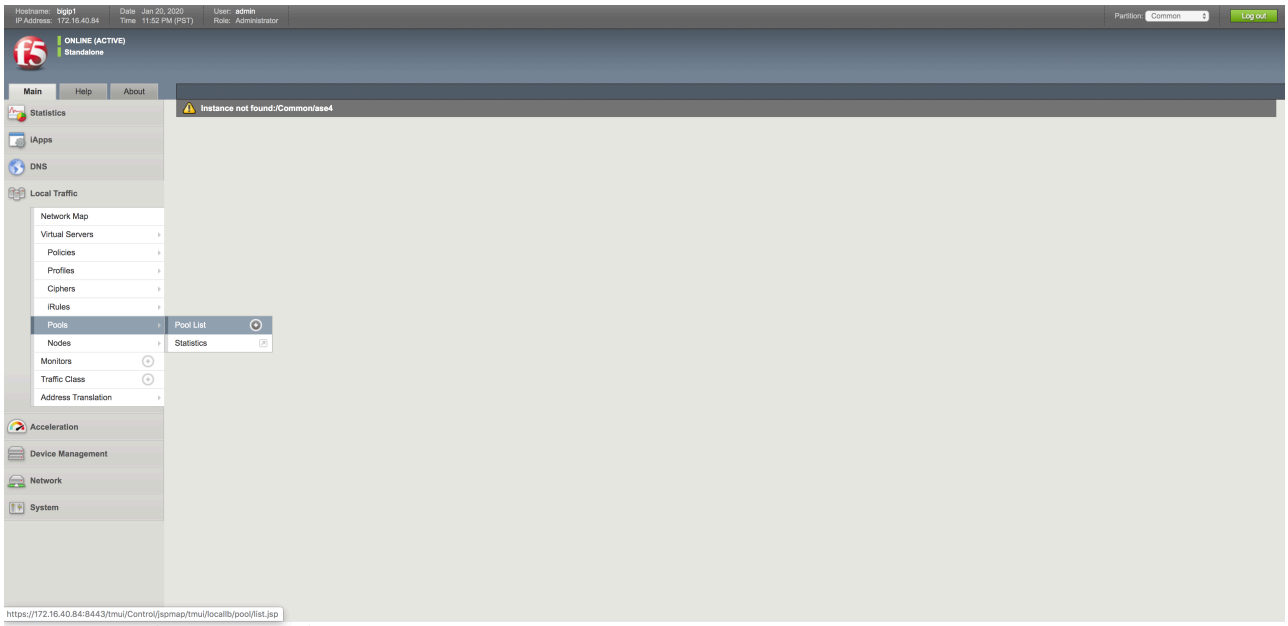
It is optional to create a backend server pool and frontend virtual server if you already have those set up. If you have existing backend server pool and frontend virtual server that you want to use, continue with the steps to .*Add PingIntelligence policy* on page 601

Complete the following steps if you do not have a backend server pool and frontend virtual server.

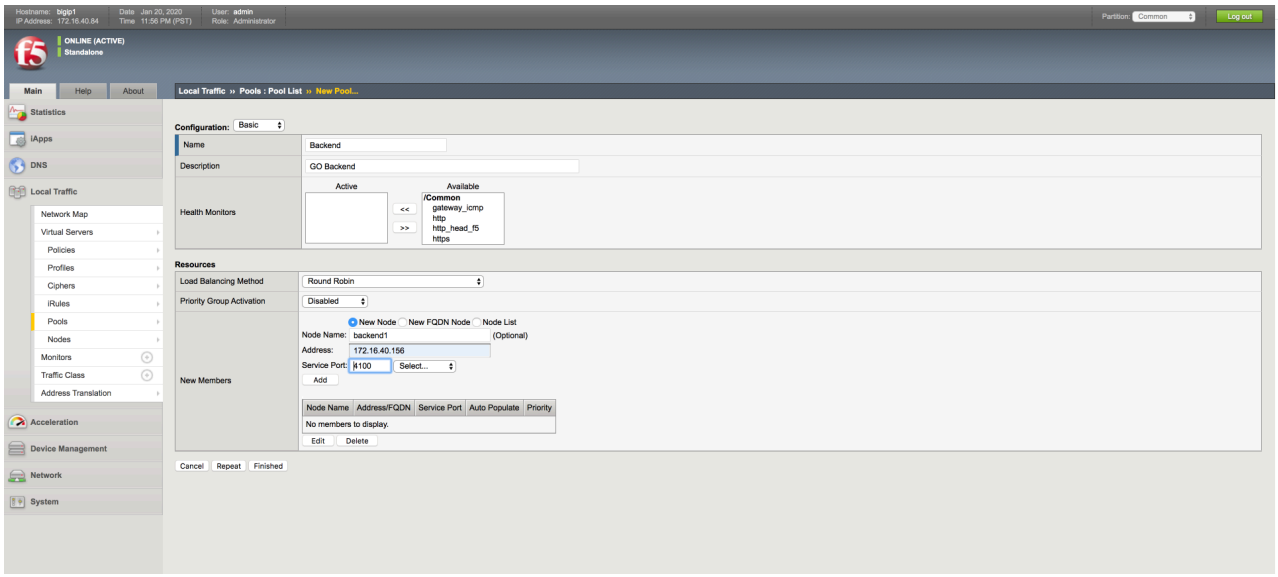
Add backend server pool

Complete the following steps to create a backend server pool:

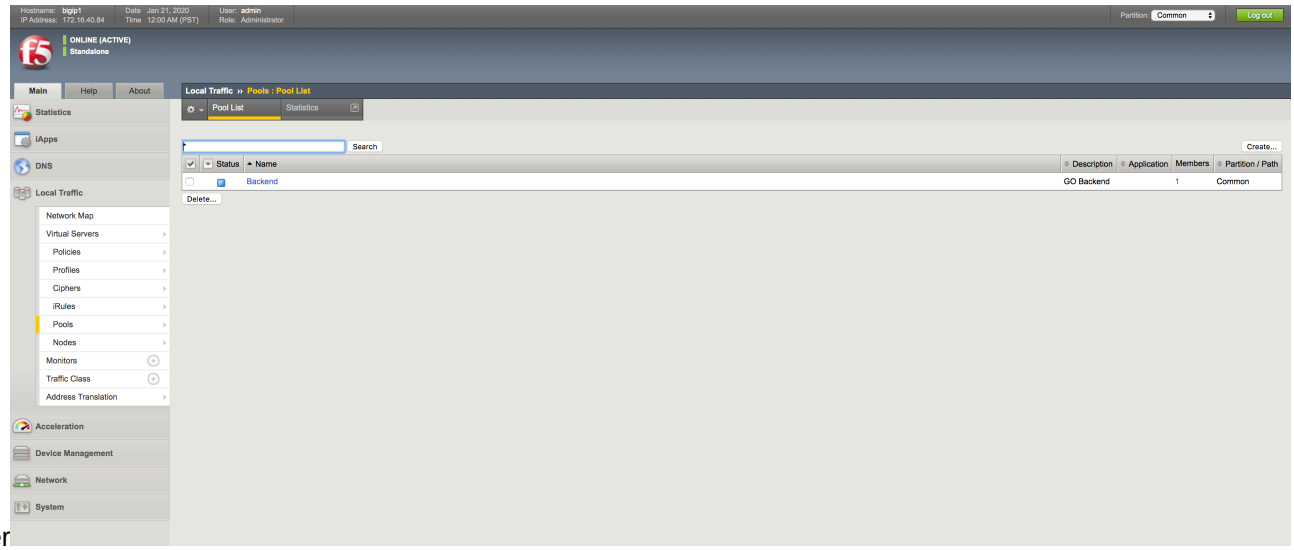
1. Navigate to **Local Traffic** > **Pools** > **Pool List** and click on Create:



2. In the configuration page for the pool, configure the fields and add a new node for the backend



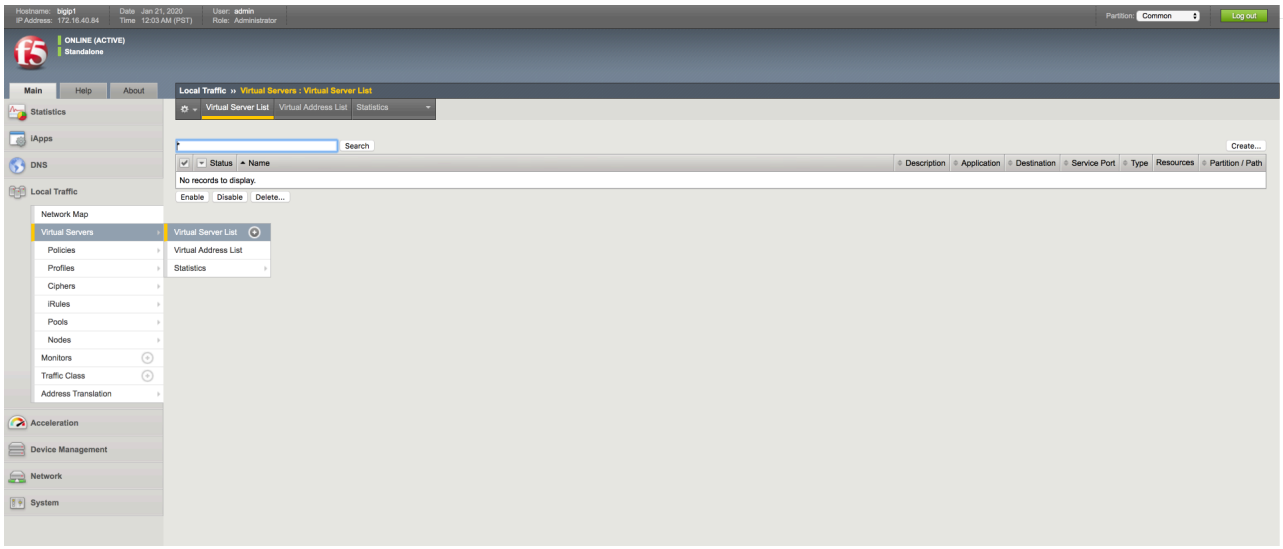
3. Click on **Finished**. This creates a backend server pool which is accessed from clients connecting to the frontend virtual server



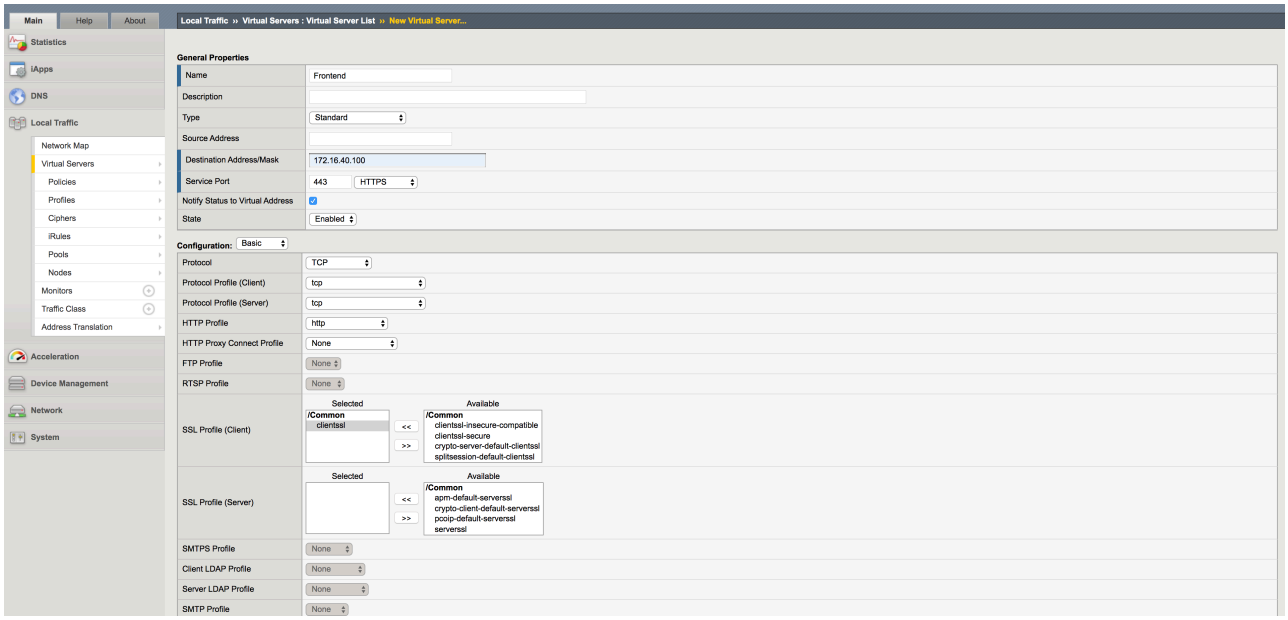
Add frontend virtual server

Complete the following steps to add a frontend virtual server:

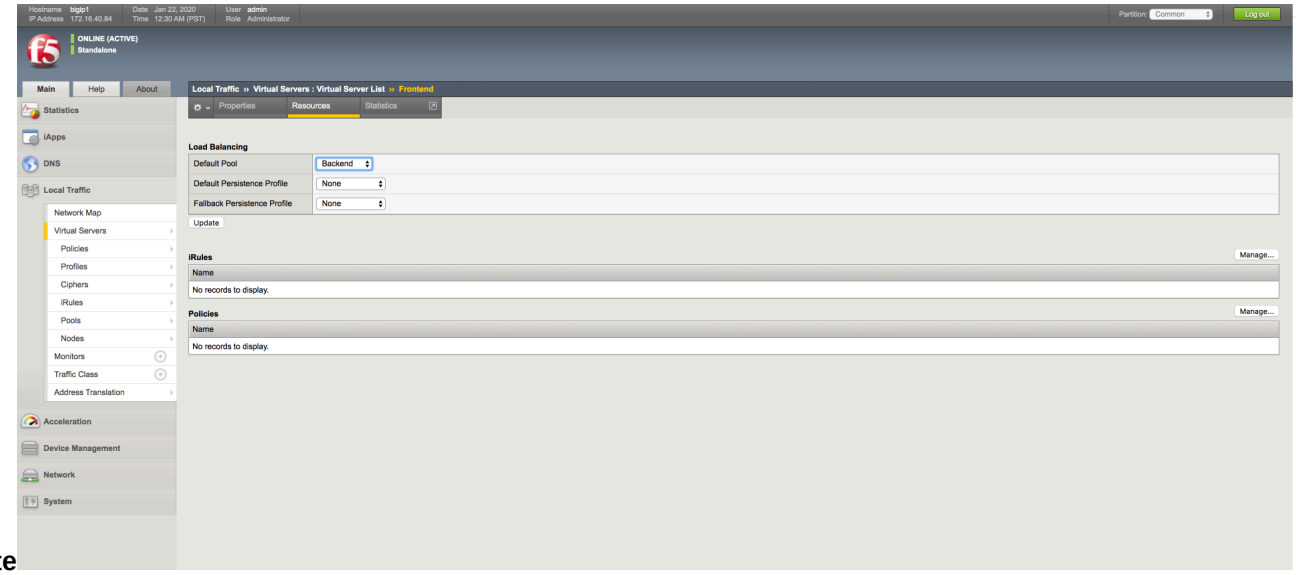
1. Navigate to **Local Traffic** > **Virtual Server** > **Virtual Server List** and click on **Create**



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 on **Finished**
4. Under **Resource** tab, add backend pool to virtual server and click on **Update**



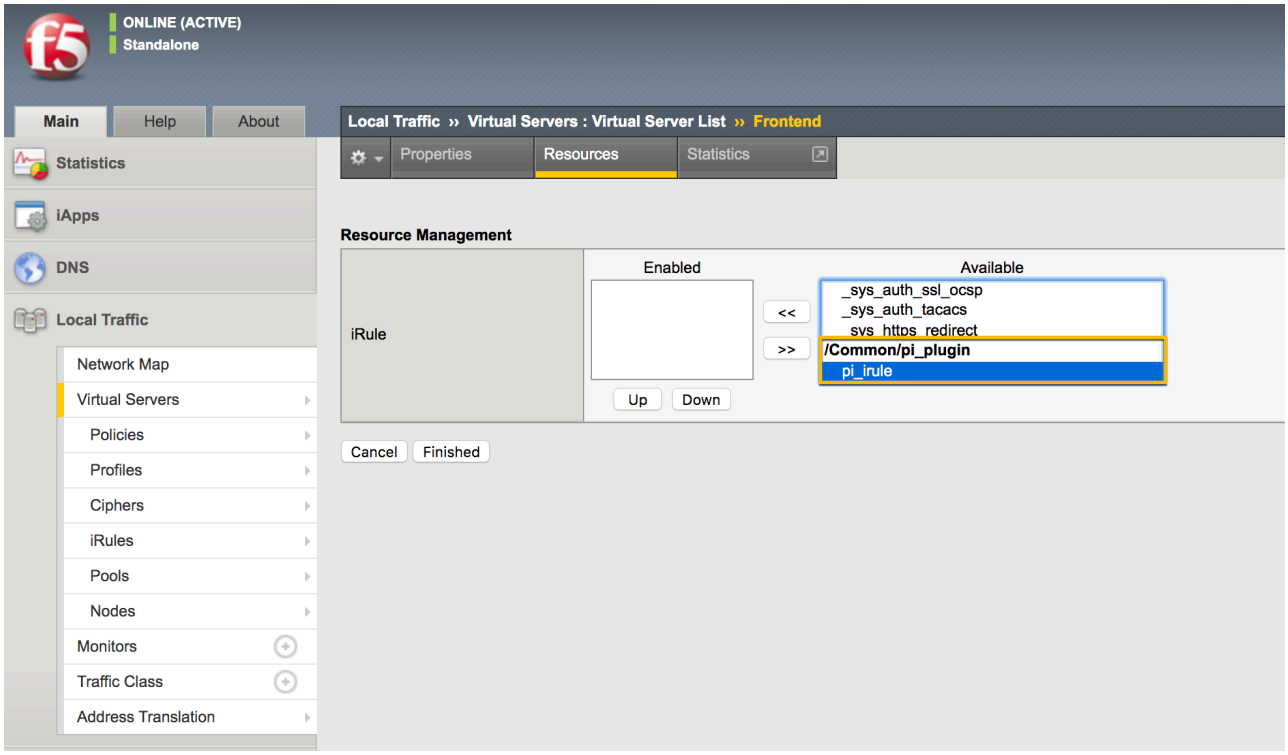
Add PingIntelligence policy

The imported PingIntelligence policy must be tied to a virtual server. Add the PingIntelligence policy to the existing or recently created virtual server..

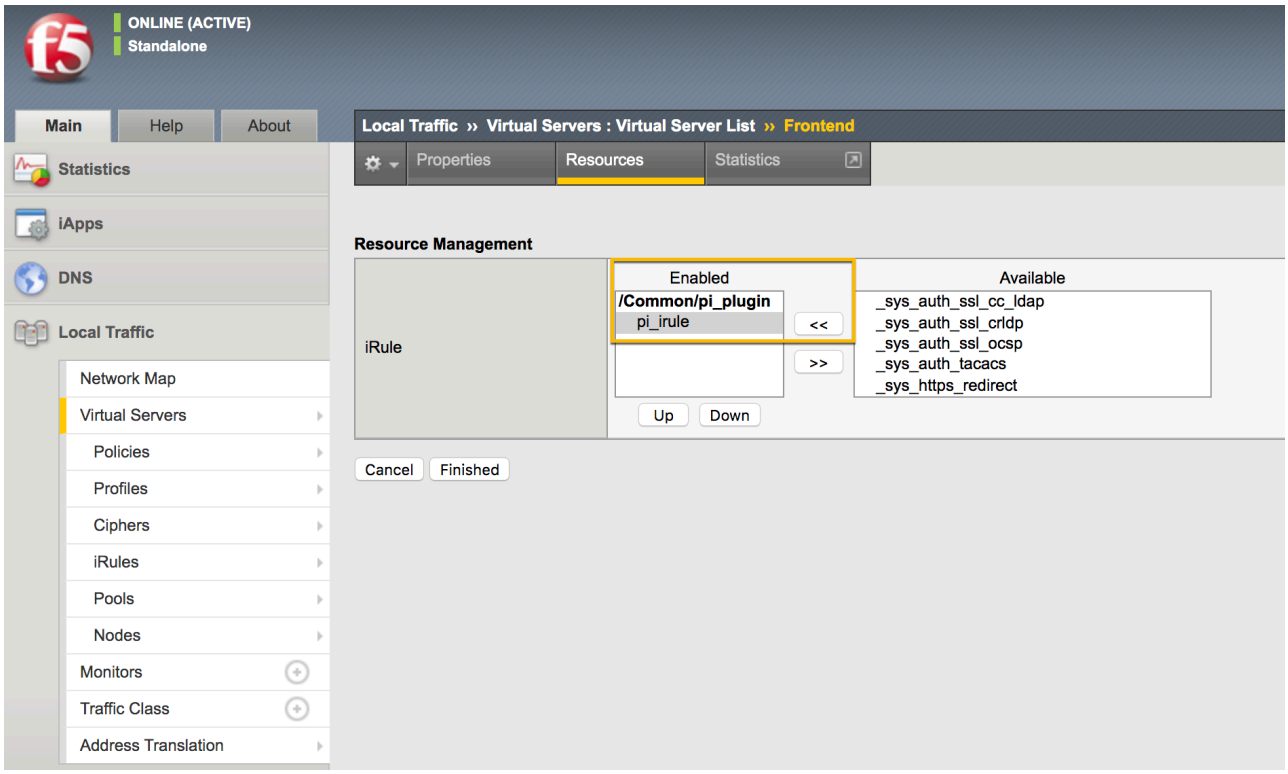
Complete the following steps to add the PingIntelligence policy to the virtual server:

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 on the **Resources** tab
4. In the **iRules** section, click on the **Manage** button.

5. Choose the iRule under the **pi_plugin** that you want to attach to the virtual server.



6. Move the pi_irule to the **Enabled** window and click on **Finished**.

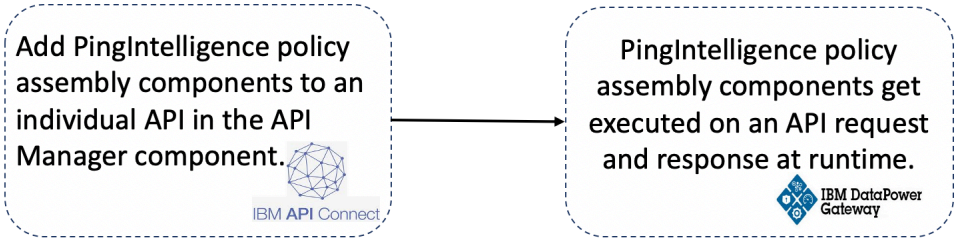


IBM DataPower gateway integration

IBM DataPower Gateway sideband integration

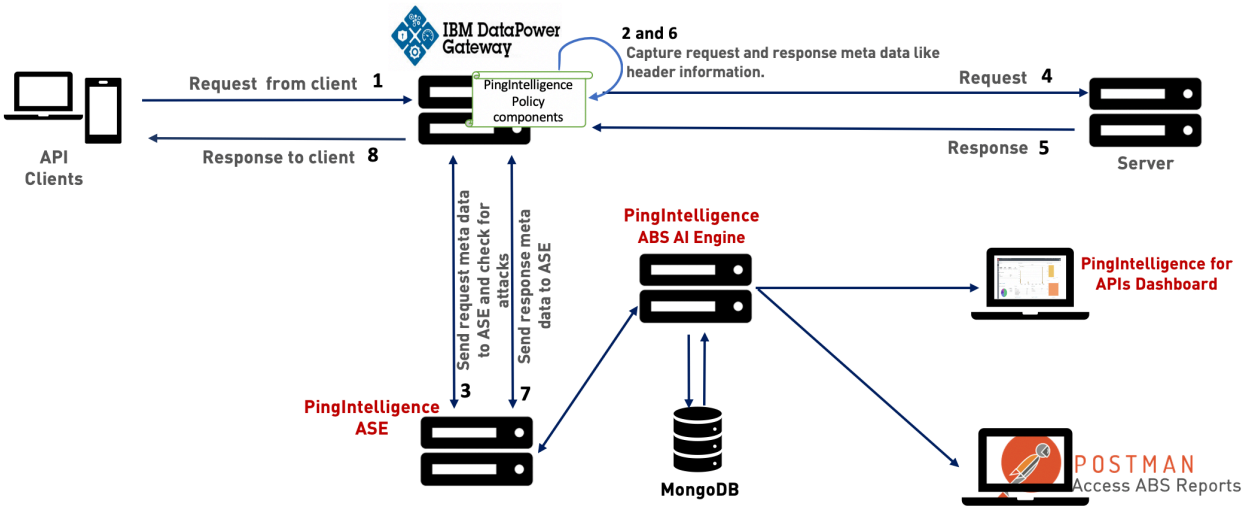
This integration guide discusses the deployment of PingIntelligence for APIs in a sideband configuration with IBM DataPower Gateway. PingIntelligence for APIs provides policy assembly components that extract the API metadata from a request or response processed by IBM DataPower Gateway. The API metadata is passed to PingIntelligence for APIs for detailed API activity reporting and attack detection. For more information on sideband deployment, see *Sideband ASE* on page 144.

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.



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. 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 Ping Intelligence 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. 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.

Prerequisites

Complete the following prerequisites before deploying the PingIntelligence policy.

Confirm the versions- The PingIntelligence policy is validated only for the following versions of IBM APIC and DataPower:

- IBM APIC v5.0.8.7
- IBM DataPower Gateway 2018.4.10

Verify User permissions- To configure PingIntelligence policy, the user must have permissions to edit and publish APIs in the API Manager.

Install PingIntelligence software- PingIntelligence software should be installed and configured. For more information on PingIntelligence deployment, see *PingIntelligence for APIs setup* on page 42 and *PingIntelligence manual deployment* on page 75.

Verify that ASE is in sideband mode- Check that 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 more information on starting ASE, see *Start and stop ASE* on page 117.

Enable sideband authentication- 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
```

Ensure SSL is configured in ASE for client side connection using self-signed certificate. For more information on configuring self-signed certificate, see *Configure SSL for external APIs* on page 126.

Enable connection keepalive between gateway and ASE- Navigate to `/opt/pingidentity/ase/config/`. Set the value of enable_sideband_keepalive to `true` in ase.conf file. If the ASE is running stop it, before making the change. Start ASE after setting the value. For more information on ASE configuration, see *ASE configuration - ase.conf* on page 146

Generate sideband authentication token- To generate the token in ASE, enter the following command in the ASE command line.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

Save the generated authentication token for further use. 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 *Configure PingIntelligence policy components* on page 608.

Deploy PingIntelligence policy

PingIntelligence for APIs provides 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 PingIntelligence policy involves:

- Step1- *Add PingIntelligence policy components* on page 605.
- Step2- *Configure PingIntelligence policy components* on page 608.

ⓘ **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.

Add PingIntelligence policy components

Complete the following steps before adding the PingIntelligence policy to your API:

1. Download the PingIntelligence policy from the *ping identity download site*.
2. Extract the policy by using the following command.

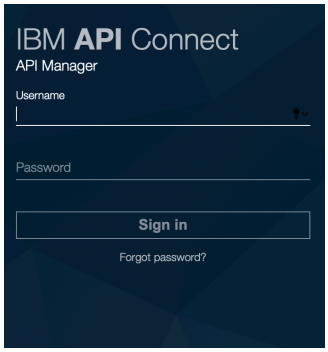
```
# tar -zxvf <<file name>>
```

For example,

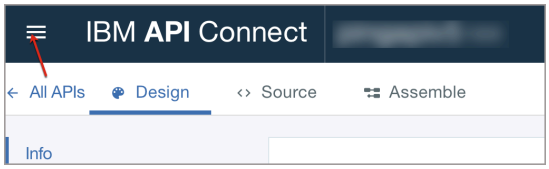
```
# tar -zxvf pi-api-ibm-policy-4.1.0.tar.gz
```

Complete the following steps to add the PingIntelligence policy components to your API in IBM API Manager:

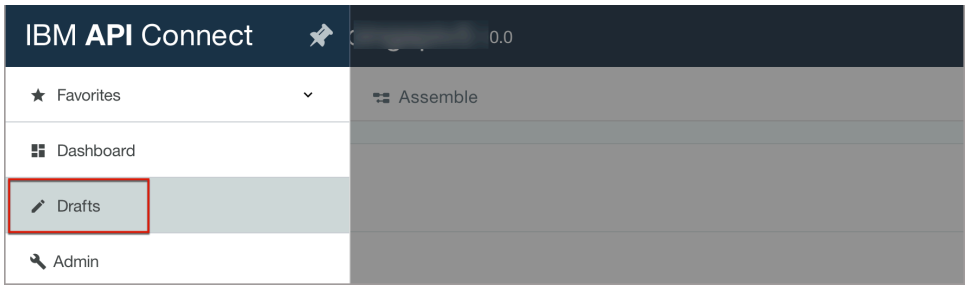
1. Log in to API Manger.



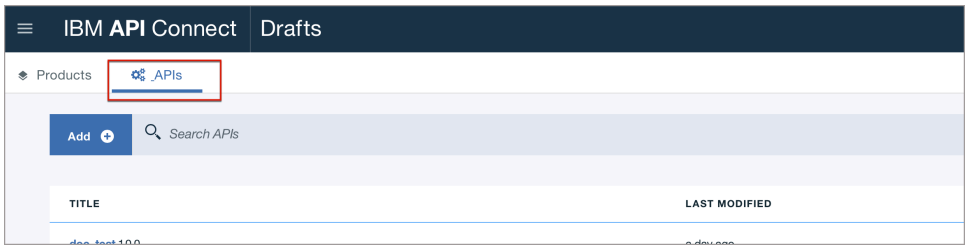
2. Click the **Menu** icon on the top-left corner, to open navigation pane.



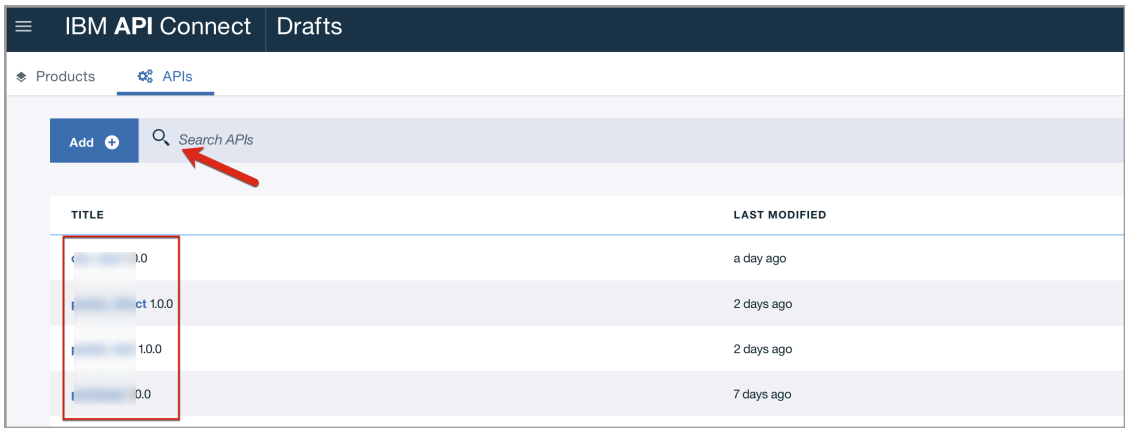
3. Click **Drafts** in the navigation pane.



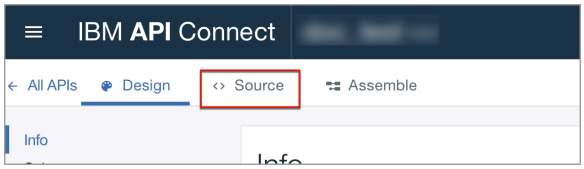
4. Click the **APIs** tab.



5. Click on your API under **TITLE** list or enter the API name in **Search APIs** dialog box and select the API.

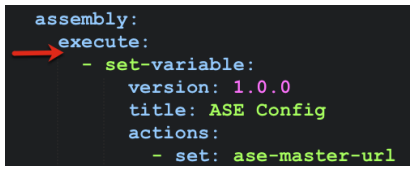


6. Click **Source** tab to edit your API definition.



7. Copy and paste the content of PingIntelligence policy into the **Assembly** block of your API definition at three places as illustrated:

   a. Open the pi_policy.yaml file, copy the content of the **set-variable:** block having **ASE Config component** and paste it in the next line after **execute:** block in your API.

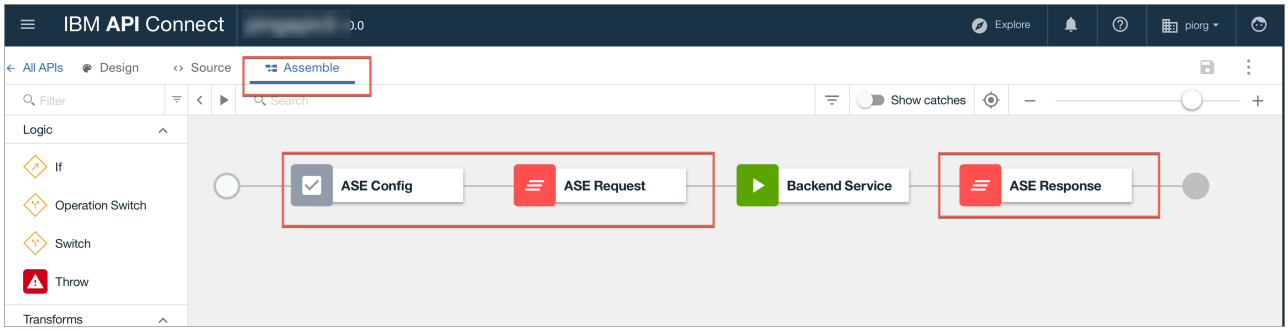


   b. Next, copy the content of the **gateway script:** block containing **ASE Request component** from pi_policy.yaml file and paste it after the immediate last line of **ASE Config component**, that was copied in step 8.1.

   c. Copy the content of the **gateway script:** block containing **ASE Response component** from 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.

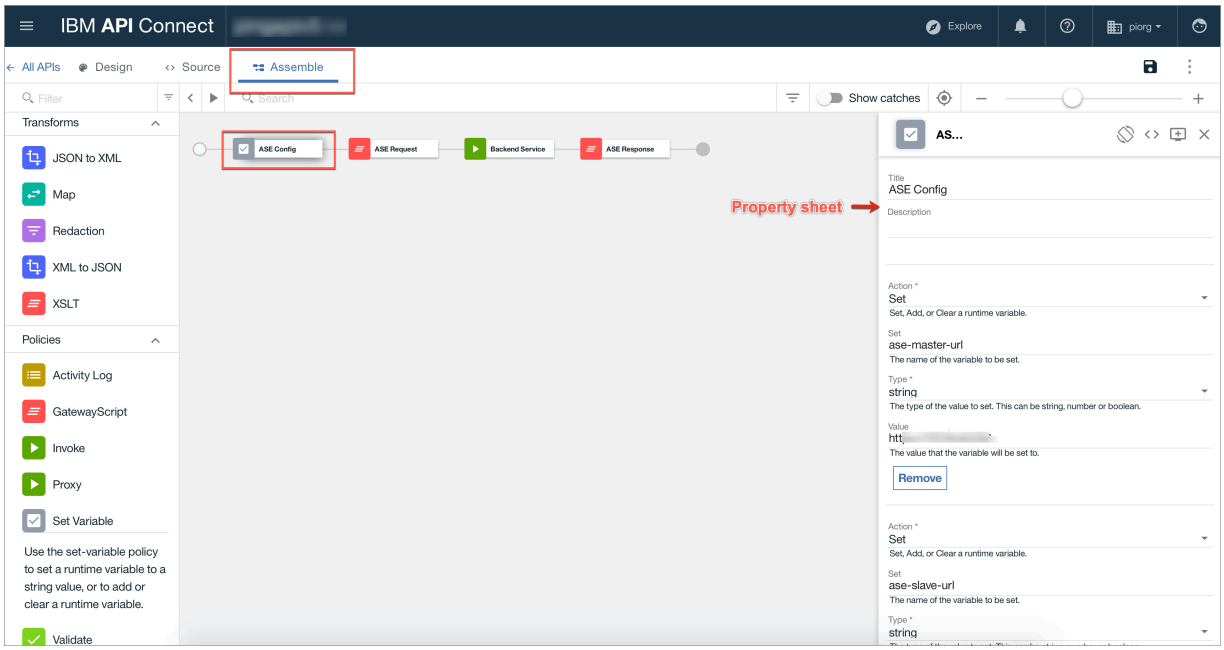
8. Click the **Validate** icon to validate your changes. Click the **Save** icon after completing the validation.
9. Click the **Assemble** tab to open the **Assemble** view. Verify the sequence of the components ASE Config, ASE Request, and ASE Response in the Policy Assembly. The order must match as highlighted in the red boxes of the following image.



Configure PingIntelligence policy components

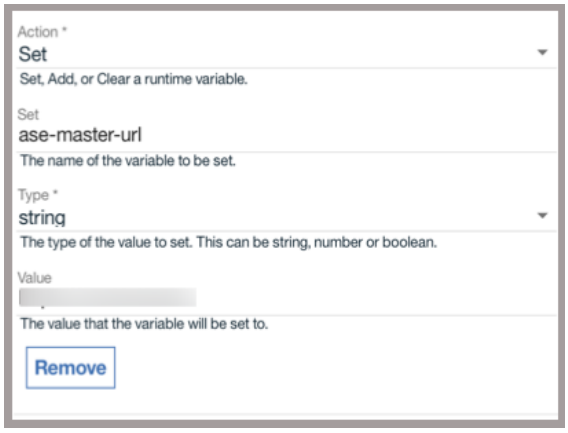
After adding the PingIntelligence policy to an API, complete the following steps to configure ASE parameters:

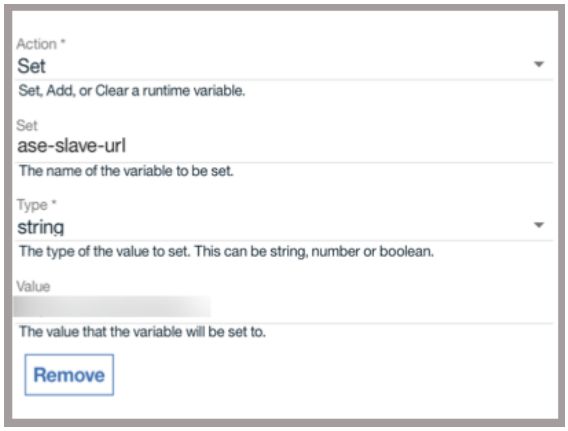
1. Click **Assemble** tab. In the main window click **ASE Config** component to open the **property sheet** on the right.

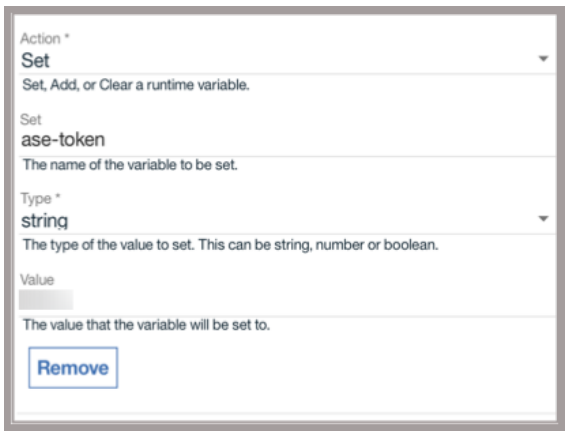


2. Configure the values for ASE master URL, ASE slave URL, and ASE token. Click the **Save** icon on the top-right corner.

> ⓘ **Note:** The following format is applicable for ASE master and slave URLs-<http/https>://<ASE-Host name or IP address>.







3. Publish your API after completing step 2 to make the PingIntelligence policy components part of your API definition.

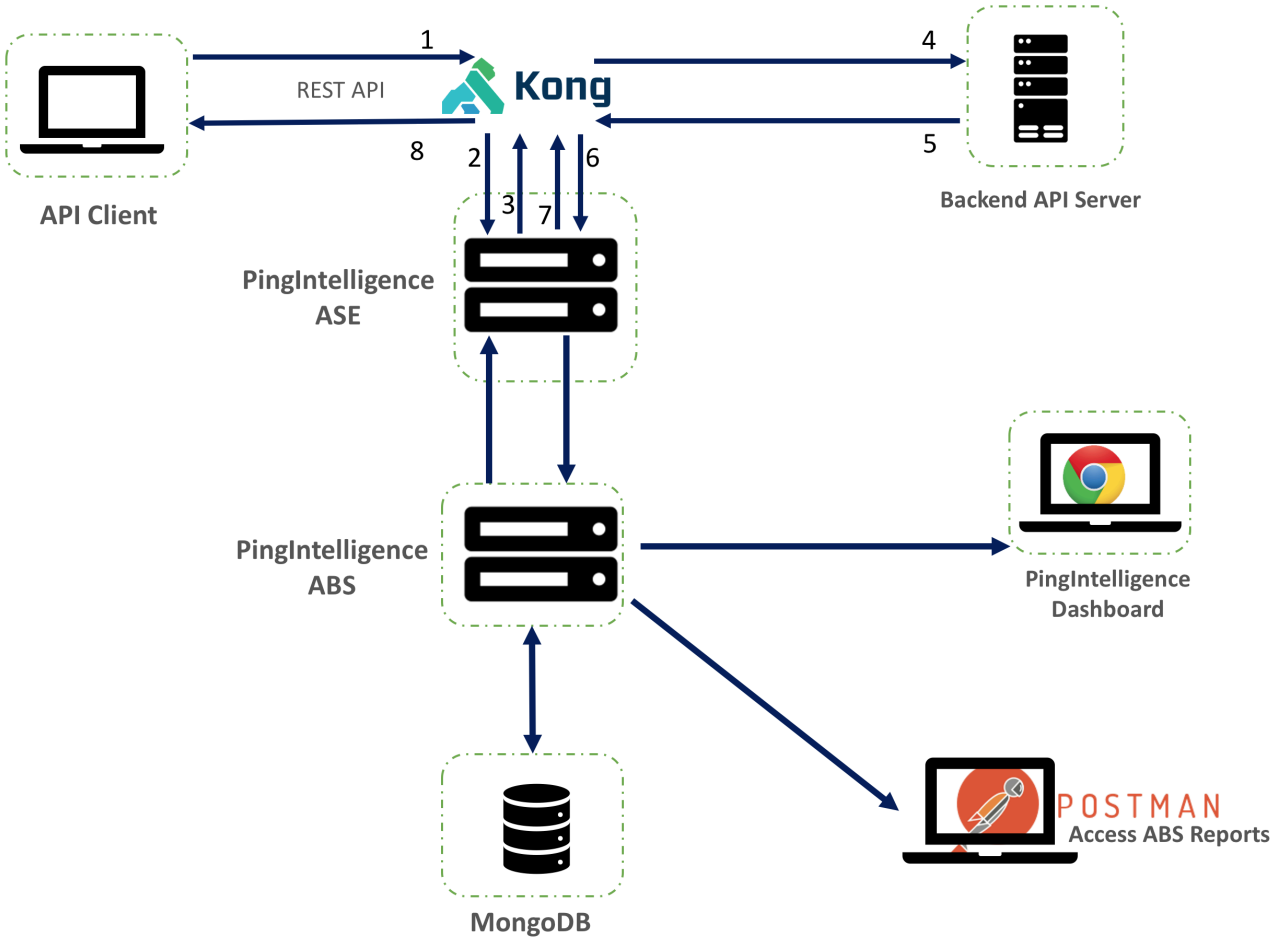
Kong API gateway integration

PingIntelligence - 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*. 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 keepalive connections.
- You can configure 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:



Here is the traffic flow through Kong API gateway and PingIntelligence for APIs components.

1. 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 Blacklist. 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.

Prerequisites

Complete the following prerequisites for PingIntelligence and Kong API gateway before deploying the PingIntelligence plugin:

PingIntelligence prerequisites

- **PingIntelligence software**: Make sure that PingIntelligence software is already installed. For more information on PingIntelligence for APIs installation, see *Automated deployment guide* or *Manual deployment guide*.
- **Verify ASE mode**: Make sure that ASE is deployed in `sideband` mode. Run the **`status`** command to check the ASE mode:

```
/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 more information on `ase.conf` file, see *ASE configuration - ase.conf* on page 146
- **Enable sideband authentication** - Enable sideband authentication if you want secure communication between Kong and ASE by entering the following command in the ASE command line:

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

  Generate sideband authentication token

  A token is required for Kong to authenticate with ASE. This token is generated in ASE and configured in the `kong.yml` file of PingIntelligence plugin. To generate the token in ASE, enter the following command in the ASE command line:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use.
- **Configure keepalive in ase.conf** - 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 keepalive paramter, see *ASE configuration - ase.conf* on page 146.

Kong Prerequisites

- Kong API gateway is already installed
- Luarocks, the Lua package manager, is installed on all the Kong nodes where you want to deploy the PingIntelligence module.

Deploy PingIntelligence policy

Complete the following steps to deploy PingIntelligence plugin for Kong API gateway:

1. *Download* the PingIntelligence plugin for Kong and copy to `/opt/` directory on all the Kong nodes where you want to deploy PingIntelligence plugin.
2. Untar the plugin file by entering the following command:

```
$ untar pi-api-kong-policy-4.1.0.tar.gz
```

3. Change directory to `/opt/pingidentity/kong-policy`

```
$ cd /opt/pingidentity/kong-policy
```

4. Run the luarocks command to deploy the PingIntelligence plugin

```
$ luarocks make *.rockspec
```

This command installs the PingIntelligence plugin files at `/usr/local/share/lua/5.1/kong/plugins/pingintelligence/` location. This location may be different based on the version of Luarocks.

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 `kong.yml` file. Following is a sample `kong.conf` file.

> ⓘ **Note:** Edit your existing kong.conf file by copying the `plugins = bundled,pingintelligence` section.

```
#------------------------------
# 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. **db-less mode**: 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` ▪ `url` ▪ `routes`	▪ `name` 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

Variable	Description
plugins: In this section, define the ASE specific variables for a service or API. ▪ `name` ▪ `service`	▪ `name`: The name of the plugin. This name was configured in `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` ▪ `ase_secondary_host` ▪ `ase_port` ▪ `ase_token` ▪ `ase_timeout` ▪ `ase_keepalive` ▪ `access_token` ▪ `use_tls` ▪ `sni_name` ▪ `tls_verify`	▪ `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 ▪ `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..

- **Apply plugin at a 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. Following is a sample `kong.yml` file:

```
#
 ------------------------------------------------------------------------------
# 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:
      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
```

- **Apply plugin at the 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.

```
#
 ------------------------------------------------------------------------------
# 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 `kong.conf` file.

Database mode

You can also optiinally configure Kong to work in the database mode. If you are running Kong in the database mode, run the `curl` command to apply the plugin at a per-service level or global level. Make sure that Kong is running when you are applying the plugin in database mode.

- **Apply plugin at 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": true,
        "sni_name": "test.ase.pi",
        "ase_port": "444",
        "ase_primary_host": "localhost",
        "ase_token": "e537d22cc0984fcfa28468066486f830",
        "ase_timeout": "5000",
        "ase_keepalive": "60000",
        "ase_secondary_host": "localhost",
        "access_token": "AccessKey",
        "use_tls": true
    }
}'
```

- **Apply plugin at the global level**: Run the following `curl` command to apply the plugin at the global level.

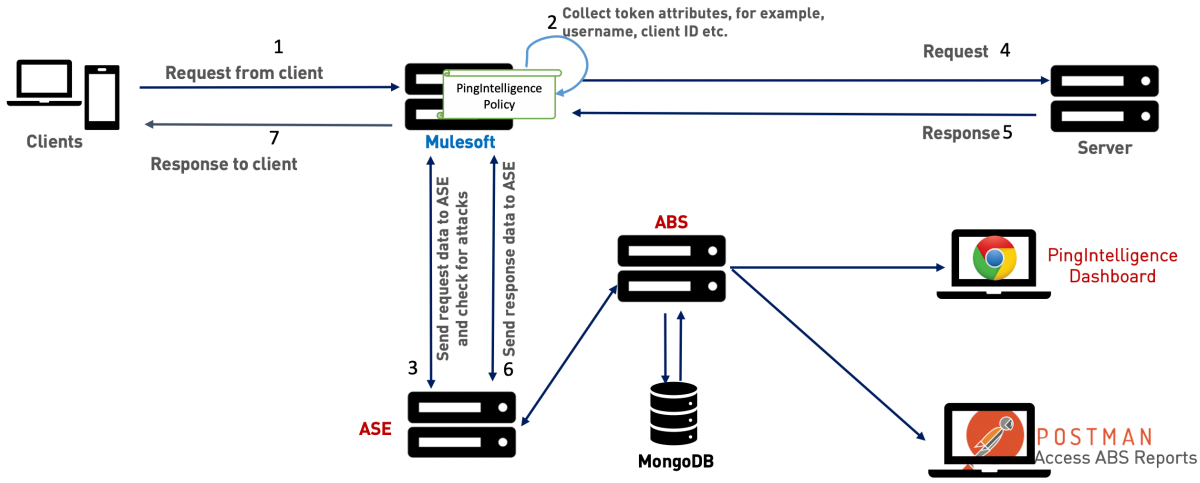
```
curl --location --request POST '<kong_ip>:<kong_admin_port>/plugins' \
--header 'Content-Type: application/json' \
--data-raw '{
 "name": "pingintelligence",
      "config": {
        "tls_verify": true,
        "sni_name": "test.ase.pi",
        "ase_port": "444",
        "ase_primary_host": "localhost",
        "ase_token": "e537d22cc0984fcfa28468066486f830",
        "ase_timeout": "5000",
        "ase_keepalive": "60000",
        "ase_secondary_host": "localhost",
        "access_token": "AccessKey",
        "use_tls": true
    }
}'
```

Mulesoft API gateway integration

Mulesoft sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with an Mulesoft API Gateway. A PingIntelligence policy is installed in the Mulesoft API Gateway and passes API metadata to PingIntelligence for detailed API activity reporting and attack detection with optional client blocking.

Traffic flow for Mulesoft integration without user information

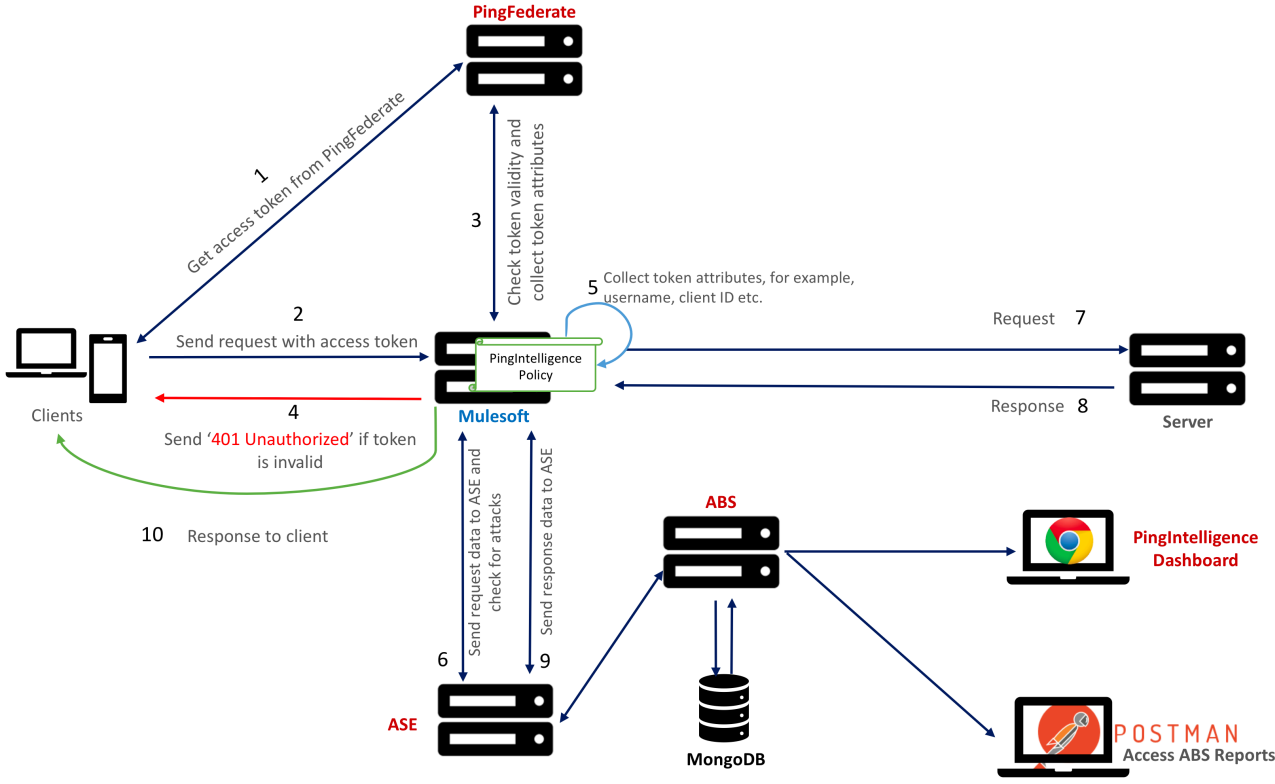


Here is the traffic flow through the Mulesoft and PingIntelligence for APIs components.

1. 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 Blacklist. 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.

Traffic flow for Mulesoft integration with user information

Here 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. Client requests and receives an access token from PingFederate.
2. Client sends a request with the access token received from PingFederate.
3. Muelsoft 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 AI generated Blacklist. 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.
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.

Prerequisites

Complete the following prerequisites before deploying PingIntelligence policy on MuleSoft:

Prerequisite:

- **Versions** The PingIntelligence policy supports the following versions of MuleSoft :
  - Version 3.9.x
  - Version 4.3.0

  If you are using any other version, contact Ping Identity support.
- **Install PingIntelligence software**

  PingIntelligence software should be installed and configured. Refer to the PingIntelligence deployment guide for your environment.
- **Verify that ASE is in sideband mode**

  Check that 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                       : 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)
```

  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.
- **Enable sideband authentication:** 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
```

- **Generate sideband authentication token**

  A token is required for Mulesoft Anypoint to authenticate with ASE. To generate the token in ASE, enter the following command in the ASE command line:

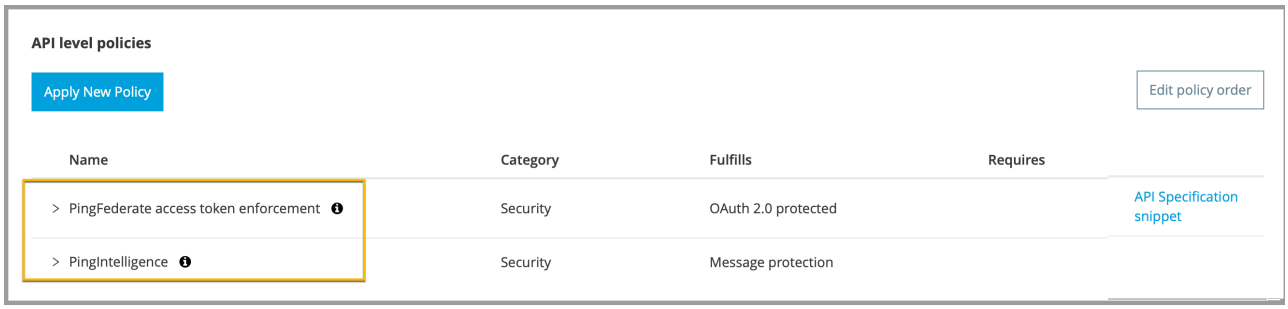
```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use.

Prerequisites to gather the user information

Complete this optional prerequisite to gather user information from PingFederate. To integrate PingFederate with Mulesoft follow the instructions detailed 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.



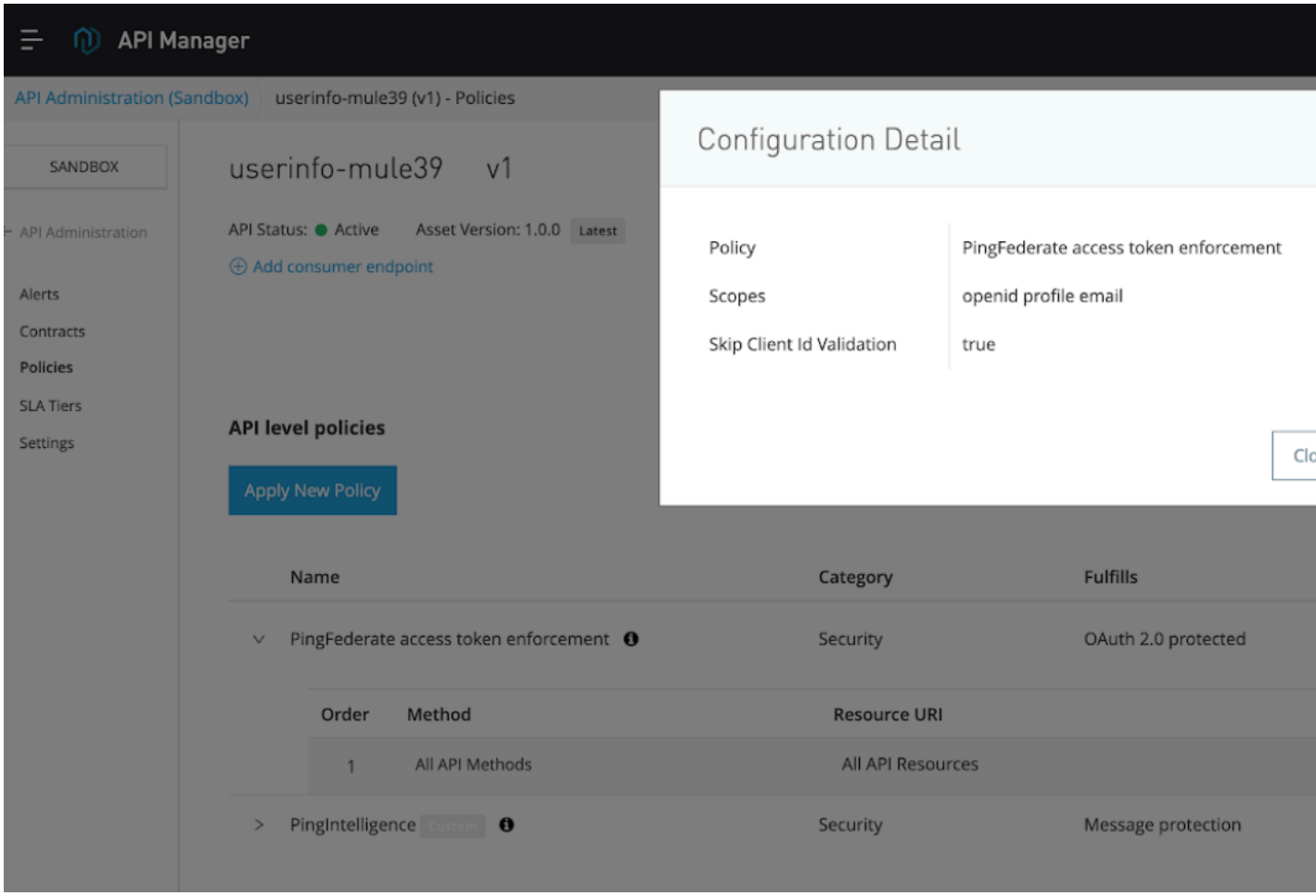
Currently the PingIntelligence policy supports PingFederate as authorization server.

Deploy PingIntelligence policy

PingIntelligence provides a policy to deploy PingIntelligence 4.0 with Mulesoft 3.9 and 4.0. The policy package has the following two files:

- `pi_policy.yaml`
- `pi_policy.xml`

Follow the steps to deploy PingIntelligence policy based on the version of 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.

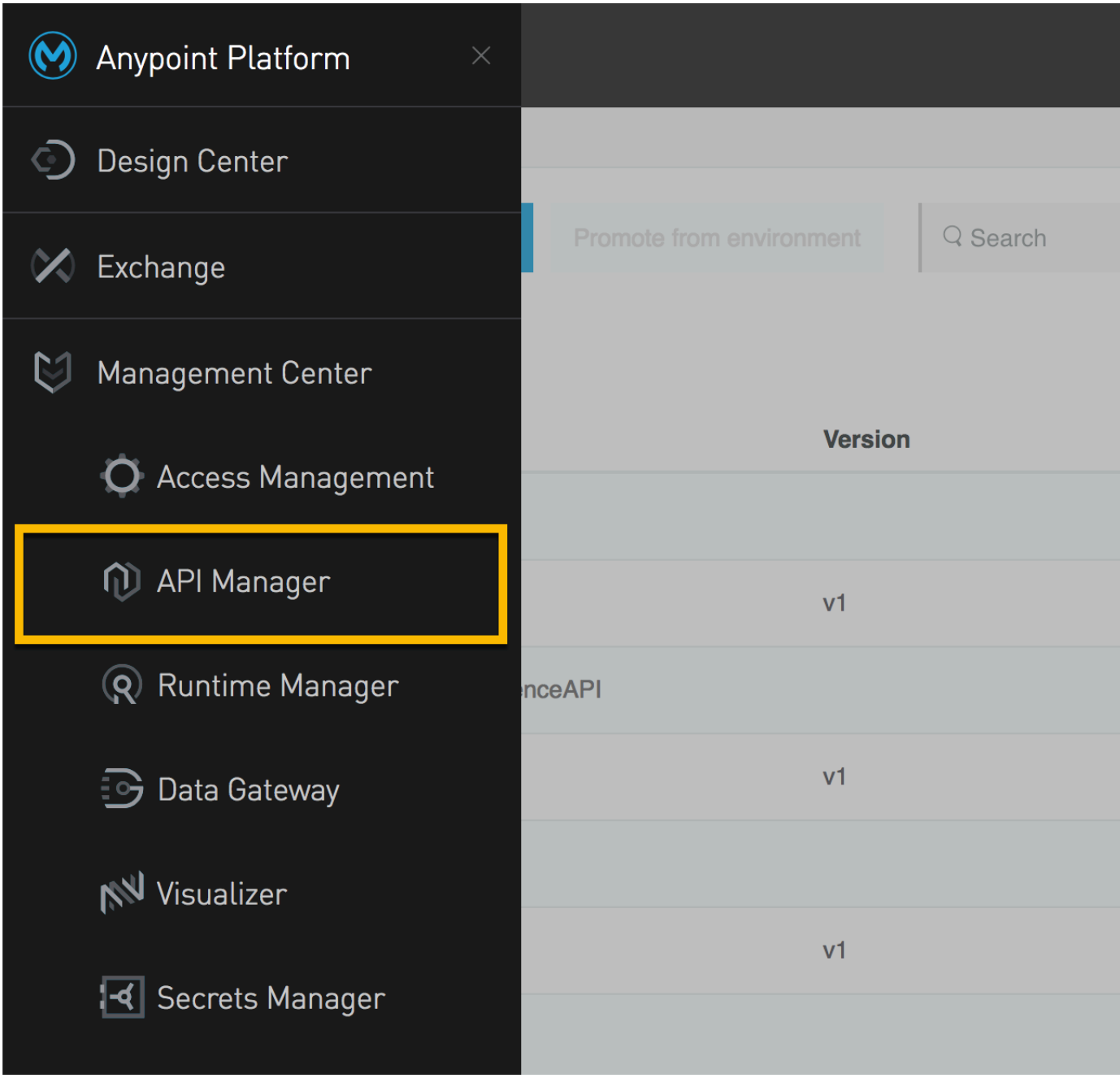


PingIntelligence for Mulesoft 3.9

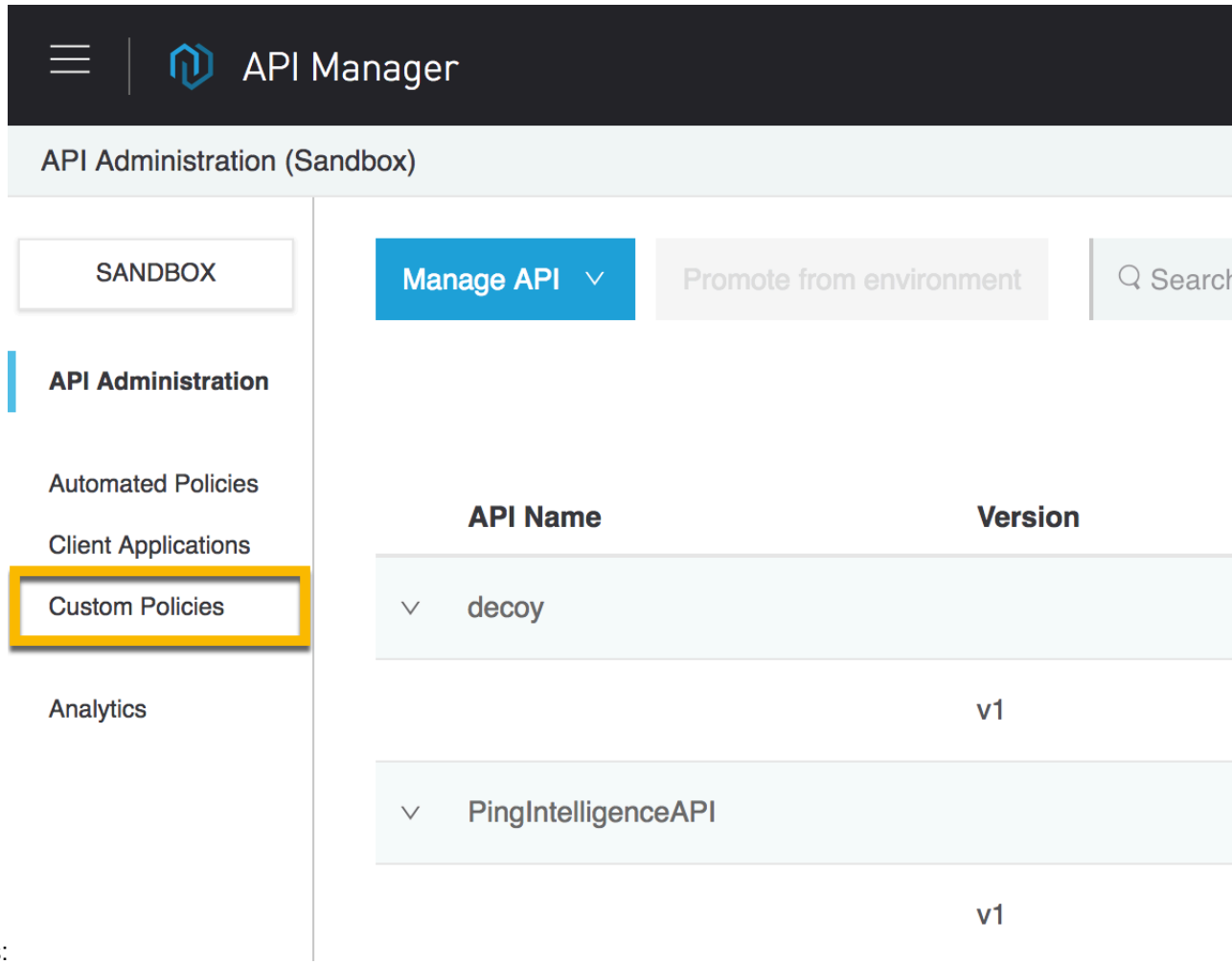
Before applying the PingIntelligence policy, make sure that the API to which you want to apply the policy is defined. The steps mentioned below use an API named PingIntelligenceAPI for illustration purpose.

Deploying PingIntelligence policy to Mulesoft Anypoint

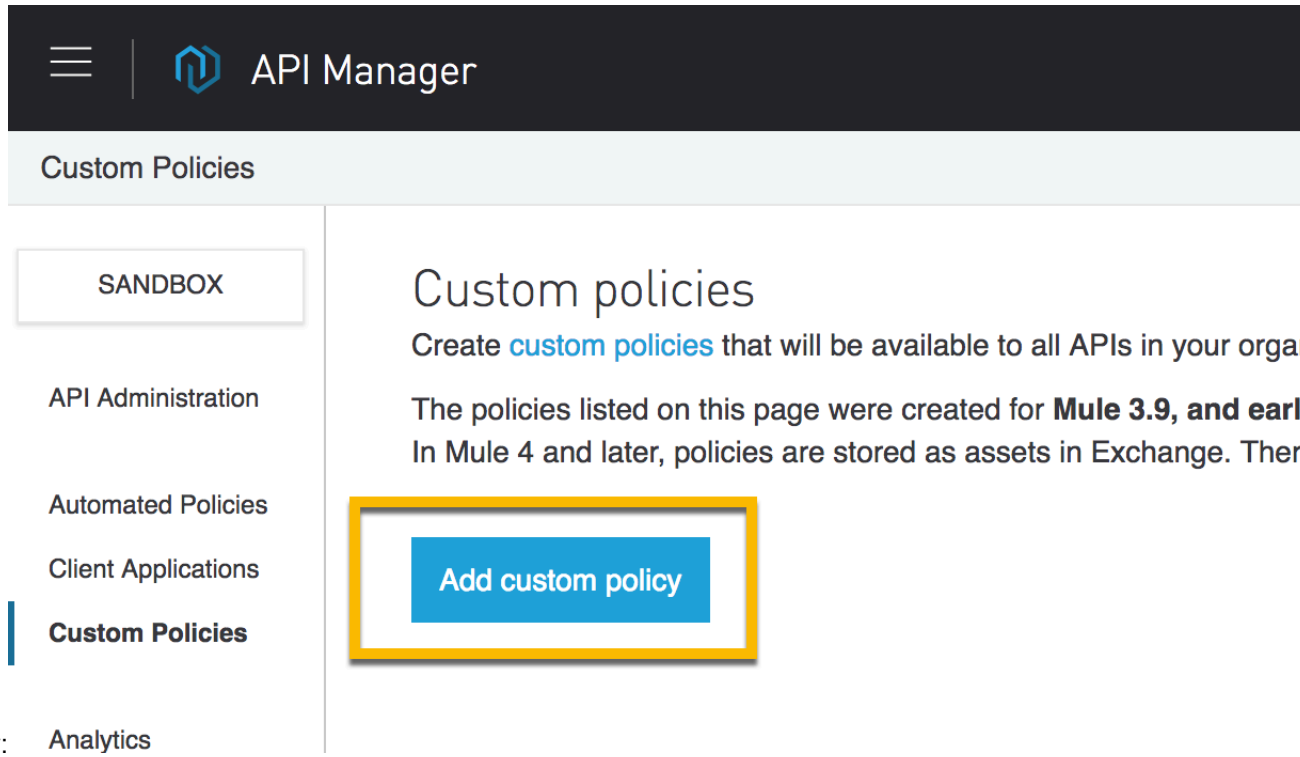
1. Login to your Mulesoft Anypoint account
2. Open API Manager by expanding the menu on the left-hand side:



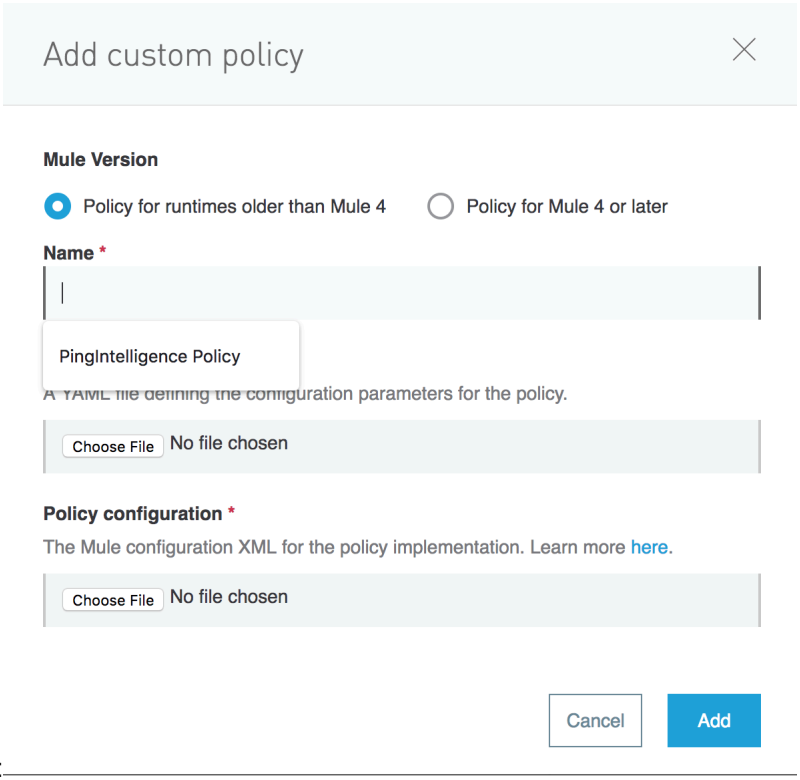
3. In the **API Administration** page, click on Custom Policies:



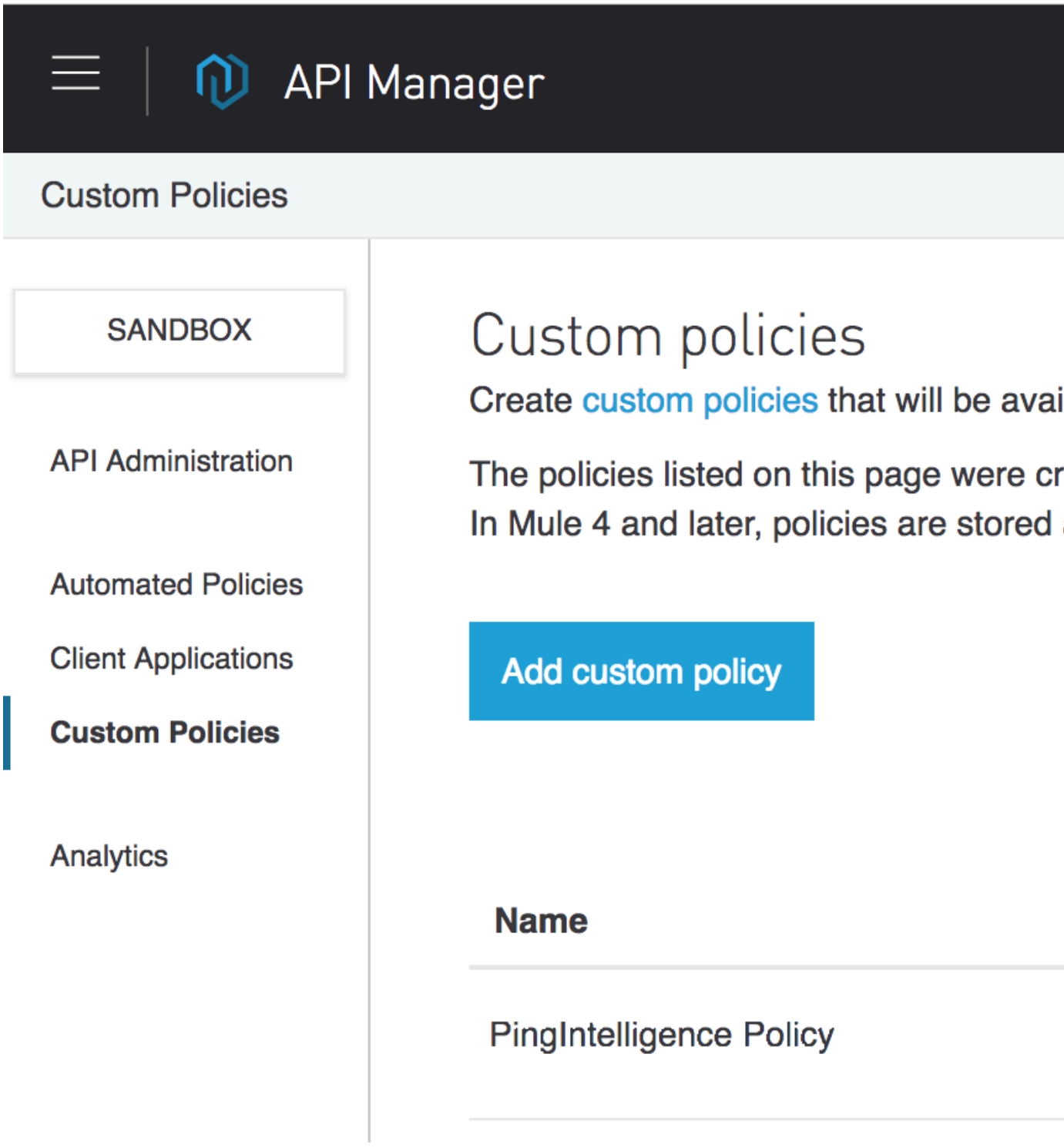
4. In the **Custom policies** page, click on **Add custom policy**:



5. In the Add custom policy pop-up window, add the policy name, for example, PingIntelligence Policy and upload the `pi_policy.yaml` and `pi_policy.xml` files:



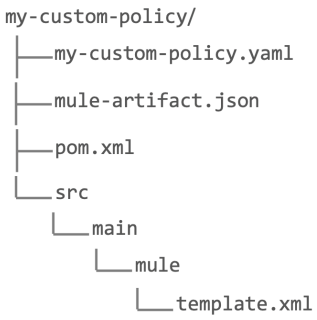
PingIntelligence Policy is added as shown below:



PingIntelligence for Mulesoft 4.x

Complete the following steps to deploy PingIntelligence policy for Mulesoft 4.x.

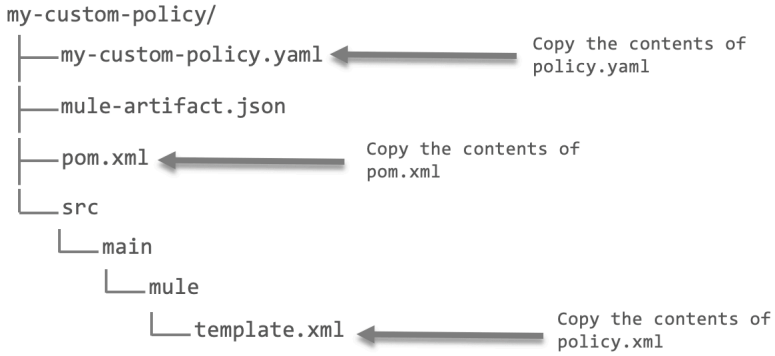
1. Create a project directory by following the instructions explained in *Getting started with Custom Policies development* link. The following screenshot shows an illustrative sample of a project directory structure.



2. 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.

   When the project's directory structure is created, replace the contents of `my-custom-policy.yaml` with that of `policy.yaml` file and the contents of `template.xml` with that of `policy.xml`. Similarly, replace the contents of `pom.xml` with that of `pom.xml` file provided in PingIntelligence policy package.



   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 under *Deploying a Policy Created Using the Maven Archetype*.

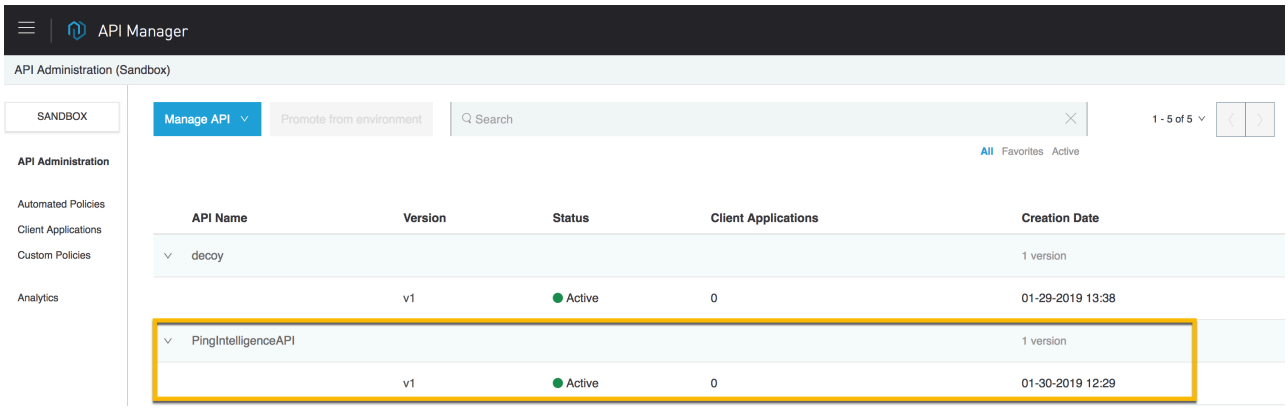
The PingIntelligence policy is now available to apply to your APIs. For more information, see *Apply PingIntelligence policy* on page 627.

Apply PingIntelligence policy

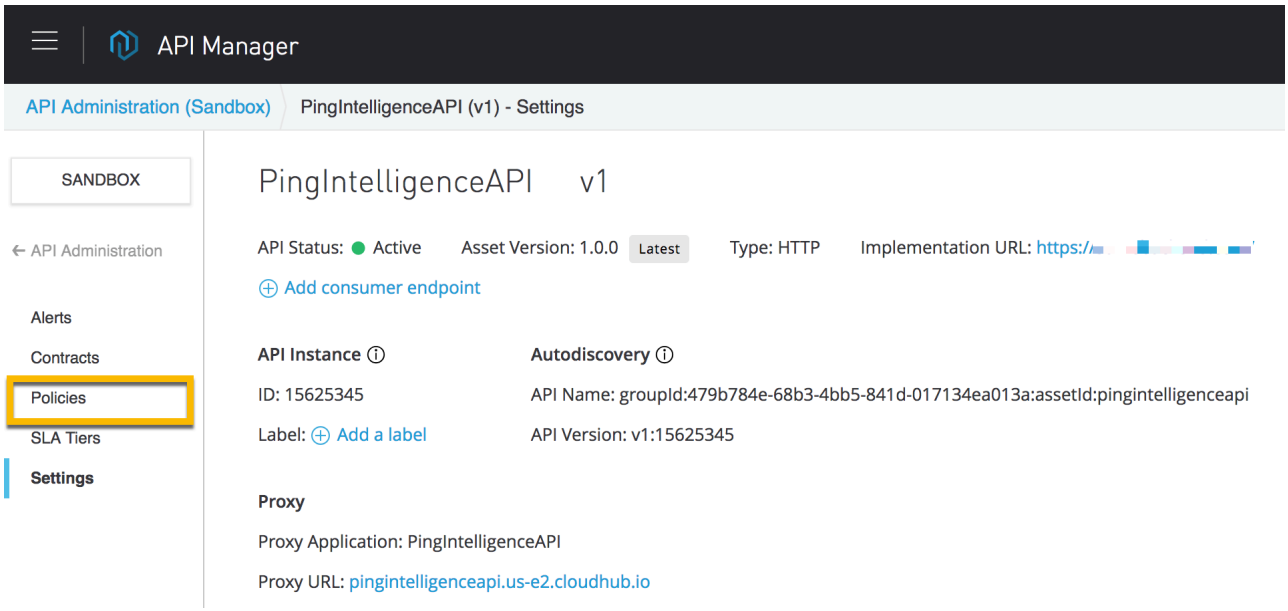
Complete the following steps to attach the PingIntelligence policy to your API:

> **Note:** At present the PingIntelligence Policy supports HTTP APIs which are configured with `Endpoint with proxy` option.

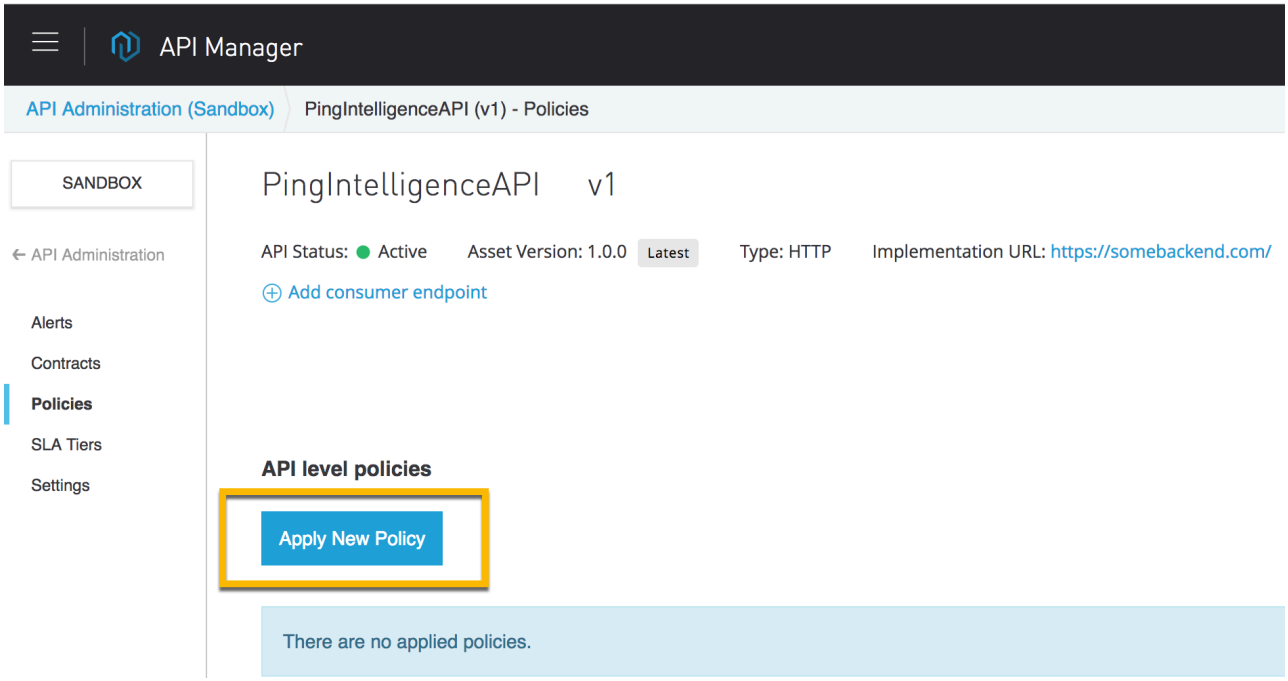
1. Navigate to the API manager and click on the **Version** of the API to which you want to attach the PingIntelligence policy:



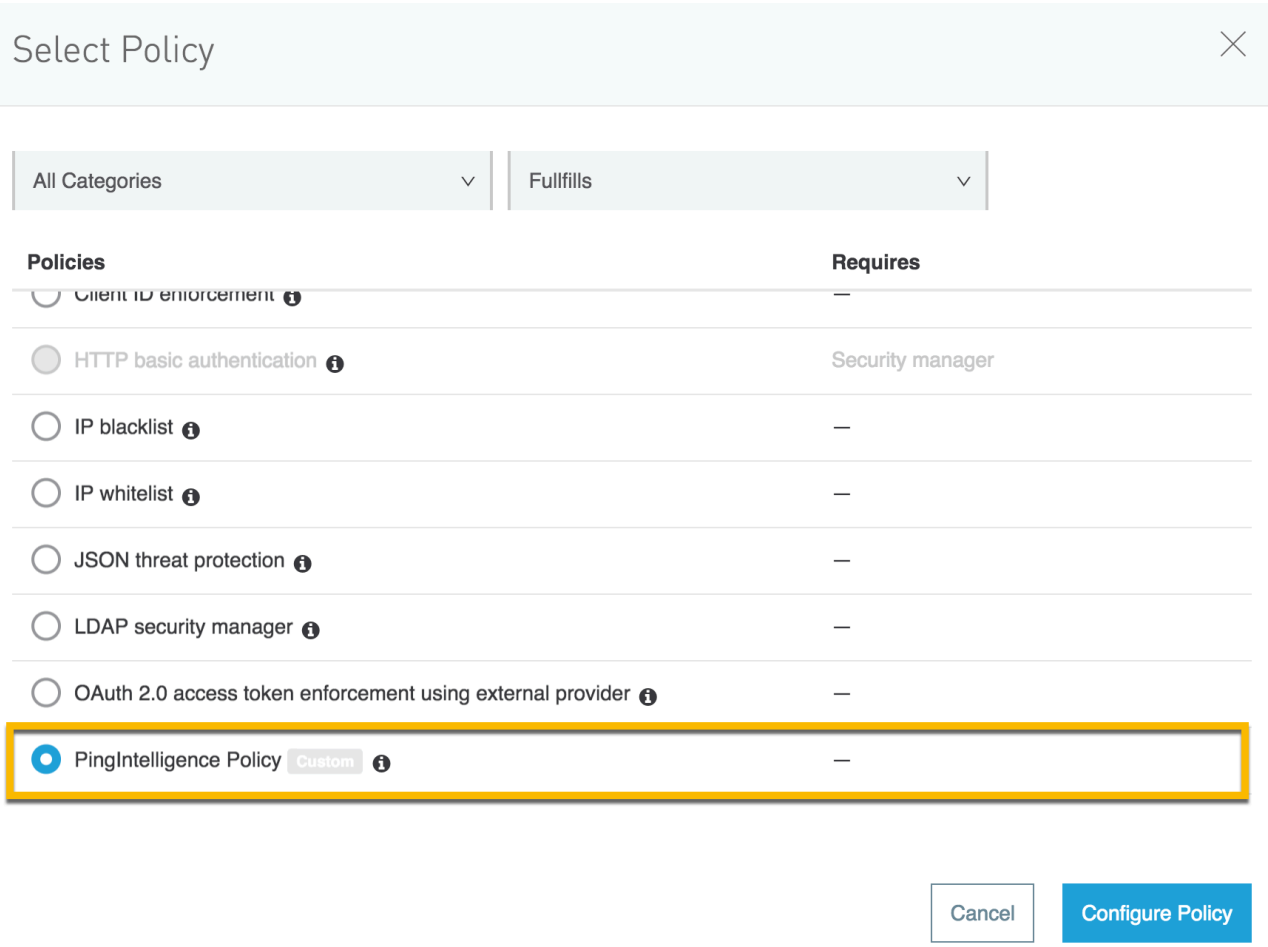
2. In the API page, click on **Policies** as shown in the following illustration:



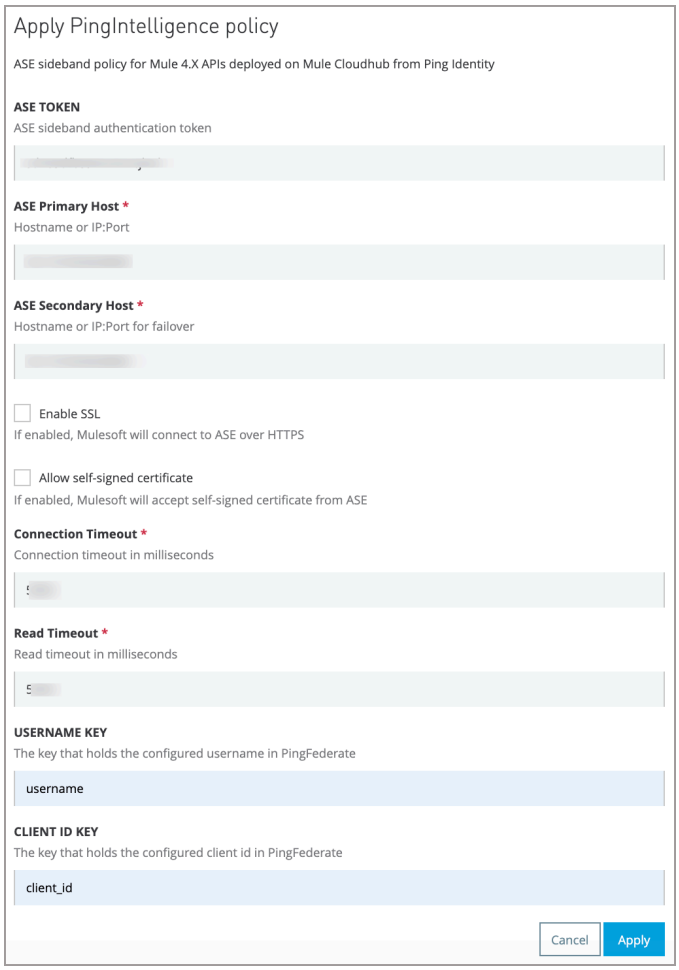
The Policies page supports applying the PingIntelligence policy to the API. Click on **Apply New**:



3. In the **Select Policy** pop-up window, select the PingIntelligence Policy and click on **Configure Policy**:

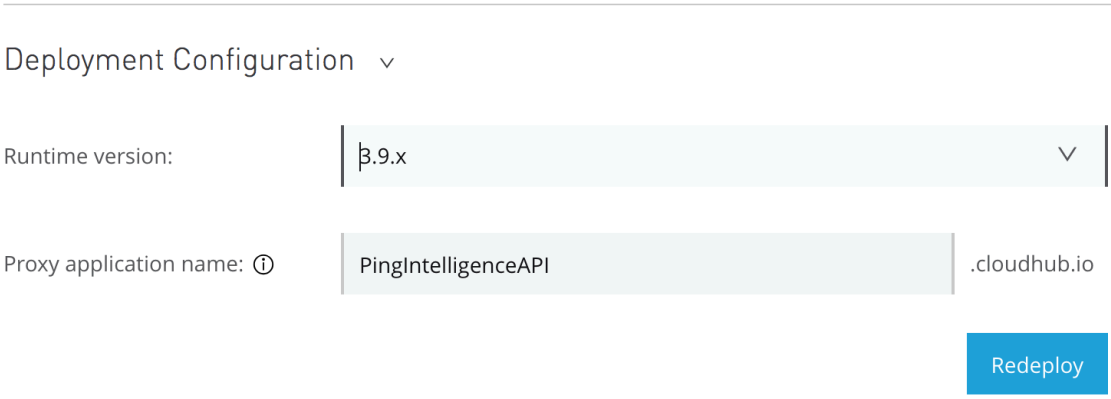


4. In the Apply policy page, enter the following values:
   - ASE Token that was generated as part of *prerequisite*
   - ASE primary and secondary host and port. The traffic is sent to the ASE secondary host only when the primary ASE node is unreachable
   - Enable SSL for a secure HTTPS connection between Mulesoft and PingIntelligence ASE
   - Check the **Allow self-signed certificate** check-box to enable Mulesoft to accept a self-signed certificate from ASE
   - Configure the **Connection Timeout** and **Read Timeout**. The default value is 5000 milliseconds (5-seconds).



> ⓘ **Note:**
>
> In the event of API Gateway not able to connect to ASE or the response from ASE is delayed, the behavior of the API gateway is governed by Connection Timeout and Read Timeout values configured in the API gateway.
>
> The Connection Timeout value governs the time the API gateway waits to establish a connection with ASE, following which it sends the client request to the backend server. Similarly, the Read Timeout value governs the time the API Gateway waits for ASE's response before sending the request to the backend server.
>
> It is a good practice to configure a small value to limit the delay in case ASE is not reachable or unresponsive.

5. Navigate to your API and click on version number as described in step-one. In the API page, scroll down to the **Deployment Configuration** section and click on



Redeploy:

> **Note:** If there are any changes to the ASE endpoints, repeat the process explained in step-five and redeploy the configurtion.

API discovery
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 JSON is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

Next steps - Integration

After the policy deployment is complete, refer the following topics for next steps:

It is recommended to read the following topics (part of the admin guides) apart from reading the *ASE* and *ABS* Admin Guides:

- *ASE port information*
- *API naming guidelines*
- Adding APIs to ASE in *Sideband ASE*. You can add individual APIs or you can configure a global API. For more information, see *API discovery and configuration* on page 321.
- *Connect ASE and ABS*

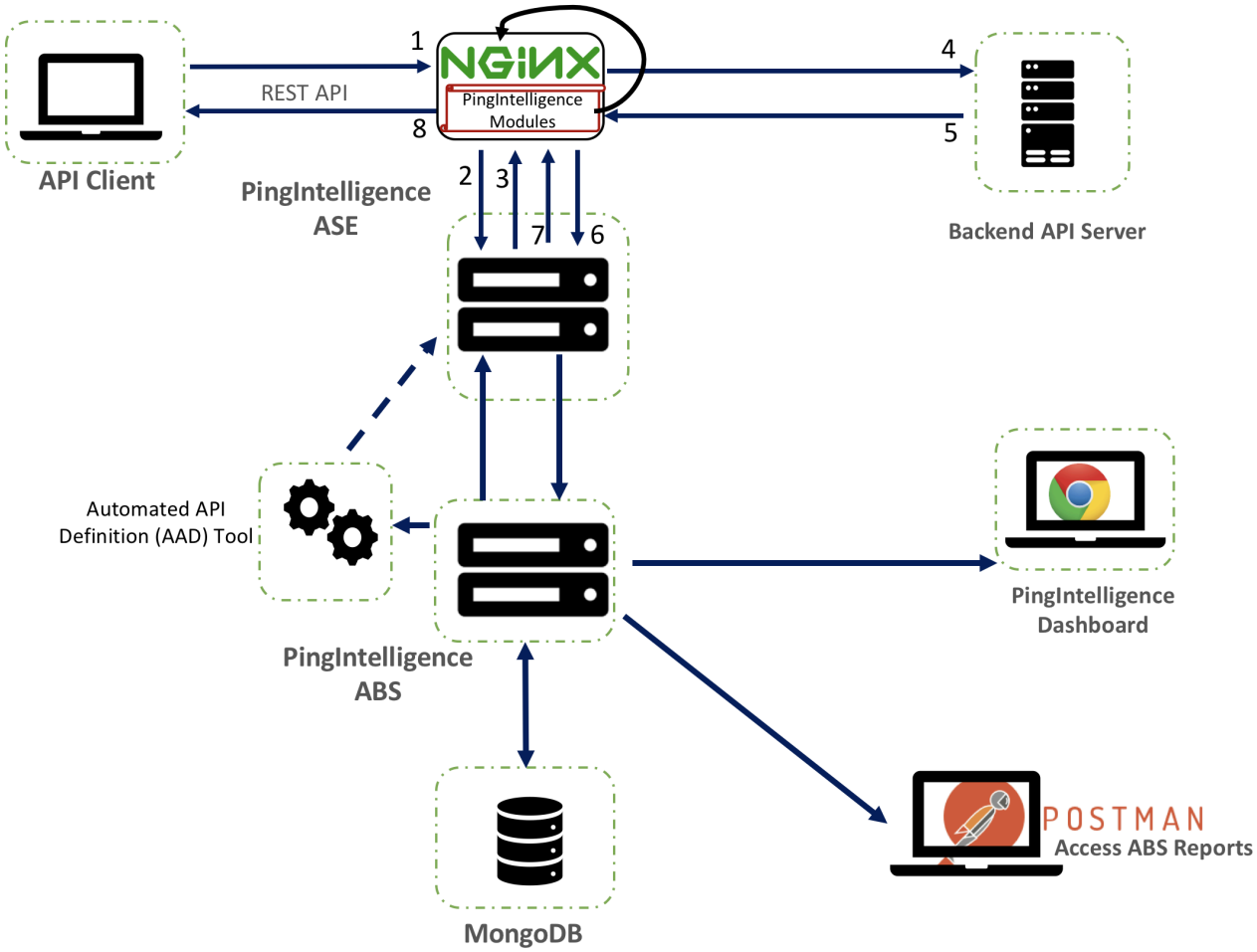
After you have added your APIs in ASE, the API model needs to be trained. The training of API model is completed in ABS. The following topics give a high level view, however it is a good practice to read the entire ABS Admin Guide.

- *Train your API model*
- *Generate and view the REST API reports using Postman*.
- *View PingIntelligence for APIs Dashboard*.

NGINX integration

NGINX sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with NGINX. 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.



Here is the traffic flow through NGINX and PingIntelligence for APIs components.

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.

Prerequisites

Prerequisite is divided in three sections. Prerequisite for PingIntelligence applies to both RHEL 7.6 and Ubuntu 16.04. Complete the prerequisite based on your operating system.

- *Prerequisites for PingIntelligence*
- *Prerequisite for RHEL 7.6*
- *Prerequisite for Ubuntu 16.04*

Prerequisite for PingIntelligence

The prerequisites are divided in the three sections:

This section assumes that you have installed and configured PingIntelligence software. For more information on PingIntelligence installation, see *PingIntelligence for APIs setup* on page 42 or *PingIntelligence manual deployment* on page 75

- **Verify that ASE is in sideband mode:** Log in 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
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.
- **Enable sideband authentication:** 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
```

- **Generate sideband authentication token**

  A token is required for NGINX to authenticate with ASE. To generate the token in ASE, enter the following command in the ASE command line:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use in *Configure NGINX for PingIntelligence* on page 636

Prerequisites for RHEL 7.6

Complete the following prerequisites before deploying PingIntelligence policy on NGINX:

- **NGINX version:** The PingIntelligence policy modules are complied for NGINX 1.14.2. If you have a different version of NGINX, contact Ping Identity support.
- **RHEL version:** RHEL 7.6. 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)
```

- **OpenSSL version:** OpenSSL `1.0.2k-fips` on your RHEL 7.6 machine. You can the check the OpenSSL version using the **`openssl version`** command.

```
$ openssl version
OpenSSL 1.0.2k-fips  26 Jan 2017
```

- **Extract ASE certificate**: Complete the following steps to 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 *Start and stop ASE* on page 117
  2. Run the following command:

     ```
     openssl s_client -connect <ASE_IP>:<ASE_PORT>  2>/dev/null </dev/null |
      sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > test.ase.pi
     ```

     This command extract the ASE certificate and appends in `test.ase.pi` file. Copy the certificate file to the NGINX machine and configure the certificate path in `nginx.conf` file.
- **Download dependencies for RHEL:** 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
  ```

ⓘ **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.

Prerequisites for Ubuntu 16.0.4 LTS

Complete the following prerequisites before deploying PingIntelligence policy on NGINX:

- **NGINX version:** The PingIntelligence policy modules are complied for NGINX 1.14.2. If you have a different version of NGINX, contact Ping Identity support.
- **Ubuntu version:** Ubuntu 16.04 LTS. 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
  ```

- **OpenSSL version:** OpenSSL `1.0.2g`. You can the check the OpenSSL version using the **`openssl version`** command:

  ```
  $ openssl version
  OpenSSL 1.0.2g  26 Jan 2017
  ```

- **Extract ASE certificate**: Complete the following steps to 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 *Start and stop ASE* on page 117
  2. Run the following command:

     ```
     openssl s_client -connect <ASE_IP>:<ASE_PORT>  2>/dev/null </dev/null |
      sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > test.ase.pi
     ```

     This command extract the ASE certificate and appends in `test.ase.pi` file. Copy the certificate file to the NGINX machine and configure the certificate path in `nginx.conf` file.
- **Download dependencies for Ubuntu:** 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
  ```

ⓘ **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.

NGINX for RHEL 7.6

To compile NGINX Community Edition 1.14.2 for PingIntelligence for APIs, complete the following 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 directory to `nginx-1.14.2`

   ```
   # cd nginx-1.14.2
   ```
4. Compile and install NGINX by running the following command: Note that 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
   ```

7. Verify the compilation by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -V
```

   The output of the above command should display `--with-compat` and `--with_http_ssl_module` flags.

Configure NGINX for PingIntelligence

Configure the `nginx.conf` setup NGINX and PingIntelligence sideband integration. Following is a summary of steps to configure NGINX for PingIntelligence:

1. Create `modules` directory inside NGINX
2. Download PingIntelligence modules
3. Copy PingIntelligence modules in the `modules` directory
4. Edit `nginx.conf` for PingIntelligence

Create `modules` directory and download PingIntelligence modules

1. Create a `modules` directory in NGINX:

```
# mkdir /usr/local/nginx/modules
```

2. Download the NGINX - PingIntelligence modules from the *download* site
3. Untar the downloaded file.

```
# tar -xvzf rhel_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
```

   The three PingIntelligence modules are:

   a. `ngx_ase_integration_module.so`
   b. `ngx_http_ase_integration_request_module.so`
   c. `ngx_http_ase_integration_response_module.so`

4. Copy the three PingIntelligence modules files for RHEL to the `modules` directory of NGINX.

```
# 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
```

Configure nginx.conf:

Complete the following steps to configure `nginx.conf` for PingIntelligence. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file at the end of the section.

1. **Load PingIntelligence modules:** Edit the `nginx.conf` file to load the PingIntelligence modules. Following is a snippet of `nginx.conf` file showing the loaded PingIntelligence modules:

```
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
```

`IP:PORT` is the IP address of primary and secondary ASE.

2. Add primary and secondary ASE hosts in `nginx.conf` in the upstream section. Following is a snippet of `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;
    }
```

3. **Configure SSL certificate:**

   Configure a SSL certificate location and ASE sideband authentication token in `nginx.conf`. ASE certificate was extracted from ASE in *Prerequisites* on page 632. 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 sideband authentication token was created as part of the *Prerequisites* in the PingIntelligence section. Following is a snippet the showing certificate location and sideband authentication token:

```
#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>;
```

> ⓘ **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.

4. **Configure ASE request and response:** Configure ASE request and response API endpoints in `nginx.conf`. 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;
```

ⓘ **Note:** `ase_integration_ssl_verify` is optional for non-SSL ASE connection.

5. **Apply PingIntelligence policy:** Apply PingIntelligence modules for APIs by configuring `location` in `nginx.conf`. `ase_integration_request` should be the first and a `ase_integration_response` should be the last.

```
location / {
       ase_integration_request;
       proxy_pass http://localhost:8080/;
       ase_integration_response;
}
```

   If you have more than more than one API, configure a `location` for each API as shown above.
6. **Verify:** 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
```

7. **Restart:** Restart NGINX by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -s stop
# sudo /usr/local/nginx/sbin/nginx
```

8. 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
```

9. Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep 4443
```

Following is a sample `nginx.conf` for reference:

```
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;
       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;
    }
}
```

NGINX for Ubuntu 16.04

To compile NGINX Community Edition 1.14.2 for PingIntelligence for APIs, complete the following 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 directory to `nginx-1.14.2`

```
# cd nginx-1.14.2
```

4. Compile and install NGINX by running the following command: Note that 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
```

7. Verify the compilation by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -V
```

The output of the above command should display `--with-compat` and `--with_http_ssl_module` flags.

Configure NGINX for PingIntelligence

Configure the `nginx.conf` setup NGINX and PingIntelligence sideband integration. Following is a summary of steps to configure NGINX for PingIntelligence:

1. Create `modules` directory inside NGINX
2. Download PingIntelligence modules
3. Copy PingIntelligence modules in the `modules` directory
4. Edit `nginx.conf` for PingIntelligence

Create **`modules`** directory and download PingIntelligence modules

1. Create a `modules` directory in NGINX:

```
# mkdir /usr/local/nginx/modules
```

2. Download the NGINX - PingIntelligence modules from the *download* 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
```

The three PingIntelligence modules are:

   a. `ngx_ase_integration_module.so`
   b. `ngx_http_ase_integration_request_module.so`
   c. `ngx_http_ase_integration_response_module.so`

4. Copy the three PingIntelligence modules for Ubuntu to the `modules` directory of NGINX.

```
# 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
```

Configure nginx.conf:

Complete the following steps to configure `nginx.conf` for PingIntelligence. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file at the end of the section.

1. **Load PingIntelligence modules:** Edit the `nginx.conf` file to load the PingIntelligence modules. Following is a snippet of `nginx.conf` file showing the loaded PingIntelligence modules:

```
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
```

   `IP:PORT` is the IP address of primary and secondary ASE.
2. Add primary and secondary ASE hosts in `nginx.conf` in the upstream section. Following is a snippet of `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;
   }
```

3. **Configure SSL certificate:**

   Configure a SSL certificate location and ASE sideband authentication token in `nginx.conf`. ASE certificate was extracted from ASE in *Prerequisites* on page 632. 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 sideband authentication token was created as part of the *Prerequisites* in the PingIntelligence section. Following is a snippet the showing certificate location and sideband authentication token:

```
#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>;
```

ⓘ **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.

4. **Configure ASE request and response:** Configure ASE request and response API endpoints in `nginx.conf`. 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;
```

ⓘ **Note:** `ase_integration_ssl_verify` is optional for non-SSL ASE connection.

5. **Apply PingIntelligence policy:** Apply PingIntelligence modules for APIs by configuring `location` in `nginx.conf`. `ase_integration_request` should be the first and a`ase_integration_response` should be the last.

```
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.
6. **Verify:** 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
```

7. **Restart:** Restart NGINX by entering the following command:

```
# sudo /usr/local/nginx/sbin/nginx -s stop
# sudo /usr/local/nginx/sbin/nginx
```

8. 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
```

9. Verify that NGINX has restarted by entering the following command:

```
# netstat -tulpn | grep 4443
```

Following is a sample `nginx.conf` for reference:

```
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;
       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;
    }
}
```

Next steps - integration

After the policy deployment is complete, refer the following topics for next steps:

It is recommended to read the following topics (part of the admin guides) apart from reading the *ASE* and *ABS* Admin Guides:

- *Customizing ASE ports* on page 115
- *API naming guidelines* on page 153
- Adding APIs to ASE in *Defining an API – API JSON configuration file* on page 153. You can add individual APIs or you can configure a global API. For more information on global API, see *API discovery and configuration* on page 321.
- *ABS AI-based security* on page 172

After you have added your APIs in ASE, the API model needs to be trained. The training of API model is completed in ABS. The following topics give a high level view, however it is a good practice to read the entire ABS Admin Guide.

- *Training the ABS model* on page 313
- *API reports using Postman* on page 386 .
- *Access PingIntelligence Dashboard* on page 17 .

API discovery

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 JSON is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

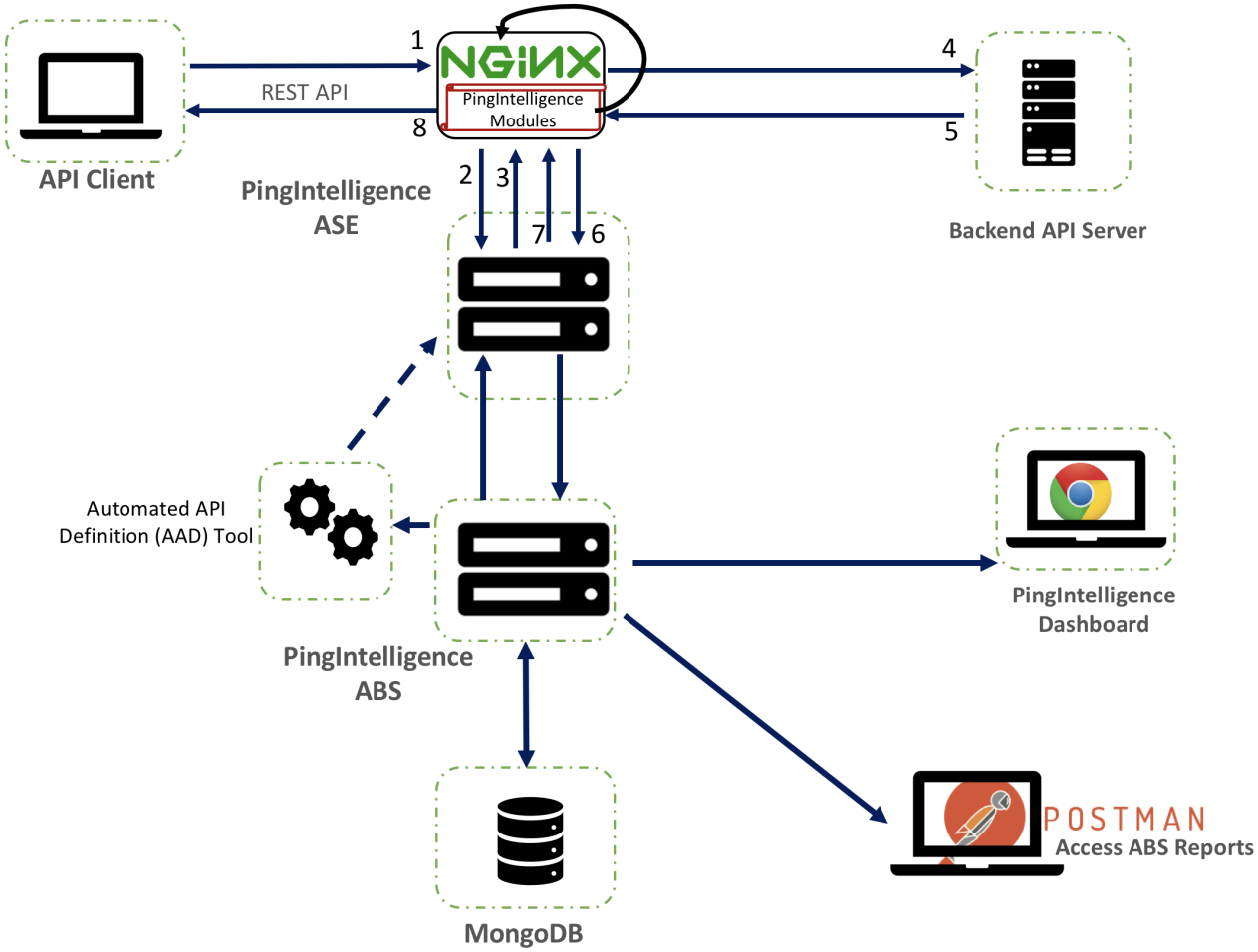
NGINX Plus integration

NGINX Plus sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with NGINX Plus. A PingIntelligence 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 4.1 software adds support for reporting and attack detection based on usernames captured from token attributes.

Traffic flow for NGINX Plus integration without user information

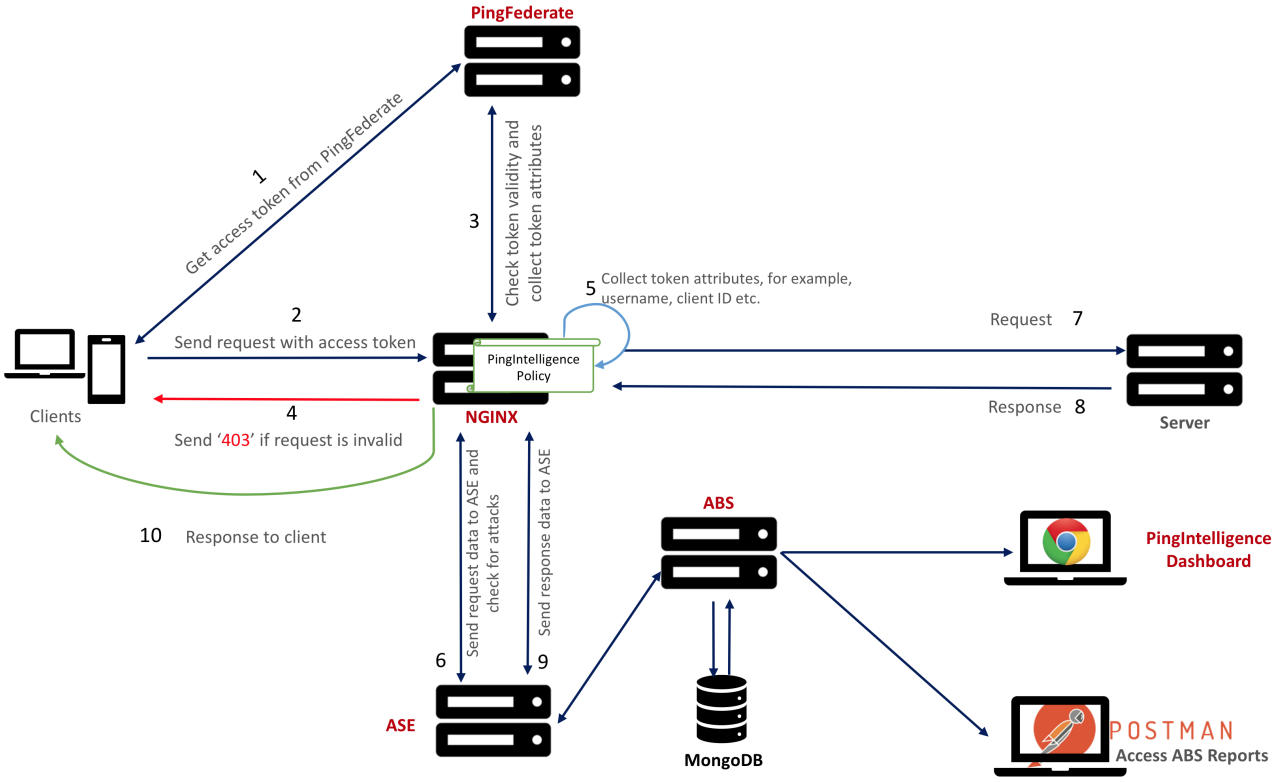
Here is the traffic flow through NGINX and PingIntelligence for APIs components.



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.

Traffic flow for NGINX Plus integration with user information

Here 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. Client requests and receives an access token from PingFederate.
2. 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 Blacklist. 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.
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.

Prerequisites

Prerequisite is divided in three sections. Prerequisite for PingIntelligence applies to both RHEL 7.6 and Ubuntu 16.0.4. Complete the prerequisite based on your operating system. The prerequisite section is divided in the following three sections:

- Prerequisite for PingIntelligence
- Prerequisite for RHEL 7.6
- Prerequisite for Ubuntu 16.04

Prerequisites for PingIntelligence

This section assumes that you have installed and configured PingIntelligence software. For more information on PingIntelligence installation, see *PingIntelligence for APIs setup* on page 42 or *PingIntelligence manual deployment* on page 75

- **Verify that ASE is in sideband mode:** Log in 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
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.
- **Enable sideband authentication:** 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
```

- **Generate sideband authentication token**

  A token is required for NGINX to authenticate with ASE. To generate the token in ASE, enter the following command in the ASE command line:

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

  Save the generated authentication token for further use in *Configure NGINX Plus for RHEL 7.6* or *Configure NGINX Plus for Ubuntu 16.0.4*

Prerequisites for RHEL 7.6

Complete the following prerequisites before deploying PingIntelligence policy on NGINX Plus:

- **NGINX Plus version:** The PingIntelligence policy modules are complied for NGINX Plus R16. If you have a different version of NGINX Plus, contact Ping Identity support.
- **RHEL version:** RHEL 7.6. 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)
```

- **OpenSSL version:** OpenSSL `1.0.2k-fips` on your RHEL 7.6 machine. You can the check the OpenSSL version 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.

- **Configure NGINX Plus certificates**: Complete the following steps to configure certificate for NGINX Plus:

  1. Create a directory for SSL certificates:

     ```
     # sudo mkdir -p /etc/ssl/nginx
     ```

  2. Login to NGINX customer portal and download `nginx-repo.key` and `nginx-repo.crt` to `/etc/ss/nginx`

  For more information, see *Installing NGINX Plus*
- **Download dependencies for RHEL**: Run the following command to download dependencies for RHEL:

```
# yum install wget ca-certificates
```

Prerequisites for Ubuntu 16.0.4

Complete the following prerequisites before deploying PingIntelligence policy on NGINX Plus:

- **NGINX version:** The PingIntelligence policy modules are complied for NGINX Plus R16. If you have a different version of NGINX Plus, contact Ping Identity support.
- **Ubuntu version:** Ubuntu 16.04 LTS. 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
```

- **OpenSSL version:** OpenSSL `1.0.2g`. You can the check the OpenSSL version using the **`openssl version`** command:

```
$ openssl version
OpenSSL 1.0.2g  26 Jan 2017
```

- **Download dependencies for Ubuntu:** Run the following command to download dependencies for Ubuntu:

```
# sudo apt-get install apt-transport-https lsb-release ca-certificates
```

- **Configure NGINX Plus certificates**: Complete the following steps to configure certificate for NGINX Plus:

  1. Create a directory for SSL certificates:

     ```
     # sudo mkdir -p /etc/ssl/nginx
     ```

  2. Login to NGINX customer portal and download `nginx-repo.key` and `nginx-repo.crt` to `/etc/ss/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.

NGINX Plus for RHEL 7.6

Complete the following steps to install NGINX Plus:

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. Complete the following steps to install Lua modules:

   a. 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
      ```

   b. Install Lua module:

      ```
      # sudo yum install nginx-plus-module-lua-16+0.10.13-1.el7_4.ngx
      ```

3. Install NGINX Plus:

   a. Check whether NGINX Plus version `nginx-plus-16-1.el7_4.ngx` is available in the list

      ```
      # sudo yum list nginx-plus --showduplicates
      ```

   b. Install NGINX Plus:

      ```
      # sudo yum install nginx-plus-16-1.el7_4.ngx
      ```

Configure NGINX Plus for PingIntelligence

Configure the `nginx.conf` to setup NGINX Plus and PingIntelligence sideband integration. Following is a summary of steps to configure NGINX Plus for PingIntelligence:

1. Create `modules` directory inside NGINX
2. Download PingIntelligence modules
3. Copy PingIntelligence modules in the `modules` directory
4. Edit `nginx.conf` for PingIntelligence

Create `modules` directory and download PingIntelligence modules

1. Create a `modules` directory in NGINX Plus:

   ```
   # mkdir /etc/nginx/modules
   ```

2. Download the NGINX Plus - PingIntelligence modules from the *download* site

3. Untar the downloaded file.

```
# tar -xvzf pi-api-nginx-plus-policy-4.1.tar.gz
modules/
modules/nginx-oss-list.txt
modules/ngx_ase_integration_module.so
modules/ngx_http_ase_integration_request_module.so
modules/ngx_http_ase_integration_response_module.so
```

   The three PingIntelligence modules are:

   - `ngx_ase_integration_module.so`
   - `ngx_http_ase_integration_request_module.so`
   - `ngx_http_ase_integration_response_module.so`

4. Copy the three PingIntelligence modules files for RHEL to the `modules` directory of NGINX Plus.

```
# 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
```

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.

Configure nginx.conf file

Complete the following steps to configure `nginx.conf` for PingIntelligence. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file at the end of the section.

1. **Load PingIntelligence modules:** Edit the `nginx.conf` file to load the PingIntelligence modules. Following is a snippet of `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
```

2. **Configure ASE primary and secondary node**: 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 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
```

3. **Configure introspect server IP address**: 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
```

4. **Configure username and client ID key**: 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
```

5. **Configure token parameter name**: 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
```

6. **Configure introspection URL**: Configure the URL of the introspection server:

```
# Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

truncated nginx.conf file
```

7. **Configure ASE Sideband token**: The sideband authentication token was created as part of the*Prerequisites* on page 648 in the PingIntelligence section. Following is a snippet showing sideband authentication token:

```
 #ASE Token for sideband authentication
    set $ase_token <ASE_TOKEN>;
```

8. **Configure ASE request and response:** Configure ASE request and response API endpoints in `nginx.conf`. Following snippet of `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
```

9. **Apply PingIntelligence policy**: You can apply PingIntelligence policy at the global level, that is, for all the APIs in your environment or for an individual API.
   - **Apply PingIntelligence policy globally**: To apply PingIntelligence policy globally, add `ase_integration_request` and `ase_integration_response` in the `server`section of `nginx.conf` as shown below:

```
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
```

   - **Apply PingIntelligence policy for a specific API**: Apply PingIntelligence modules for APIs by configuring `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 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
```

10. **Verify syntactical correctness of nginx.conf**: 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
```

Final configuration steps: Complete the following steps to configure 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.

```
# 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
```

Following is a sample `nginx.conf` file:

```
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")';
    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 ~* ([^\s]+)(\s+)([^\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;
```

```
        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;
    }
}

}
```

NGINX for Ubuntu 16.0.4

Complete the following steps to install NGINX Plus:

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. Complete the following steps to install NGINX Plus with Lua modules:
   a. Check whether `16-1~xenial` is available in the list

```
# sudo apt-cache show nginx-plus | grep "Version"
```

   b. 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
```

Configure NGINX Plus for PingIntelligence

Configure the `nginx.conf` to setup NGINX Plus and PingIntelligence sideband integration. Following is a summary of steps to configure NGINX Plus for PingIntelligence:

1. Create `modules` directory inside NGINX working directory
2. Download PingIntelligence modules
3. Copy PingIntelligence modules in the `modules` directory
4. Edit `nginx.conf` for PingIntelligence

Create `modules` directory and download PingIntelligence modules

1. Create a `modules` directory in NGINX Plus:

```
# mkdir /etc/nginx/modules
```

2. Download the NGINX Plus - PingIntelligence modules from the *download* site
3. Untar the downloaded file.

```
# tar -xvzf pi-api-nginx-plus-policy-4.1.tar.gz
modules/
modules/nginx-oss-list.txt
modules/ngx_ase_integration_module.so
modules/ngx_http_ase_integration_request_module.so
modules/ngx_http_ase_integration_response_module.so
```

   The three PingIntelligence modules are:

   - `ngx_ase_integration_module.so`
   - `ngx_http_ase_integration_request_module.so`
   - `ngx_http_ase_integration_response_module.so`

4. Copy the three PingIntelligence modules files for RHEL to the `modules` directory of NGINX Plus.

```
# 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
```

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.

Configure nginx.conf file

Complete the following steps to configure `nginx.conf` for PingIntelligence. Make sure that the PingIntelligence module and other configurations are added at the correct place in `nginx.conf` as shown in the sample file at the end of the section.

1. **Load PingIntelligence modules:** Edit the `nginx.conf` file to load the PingIntelligence modules. Following is a snippet of `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
```

2. **Configure ASE primary and secondary node**: 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 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
```

3. **Configure introspect server IP address**: 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
```

4. **Configure username and client ID key**: 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
```

5. **Configure token parameter name**: 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
```

6. **Configure introspection URL**: Configure the URL of the introspection server:

```
# Set introspection URL
    set $oauth_url https://introspect_server/as/introspect.oauth2;

truncated nginx.conf file
```

7. **Configure ASE Sideband token**: The sideband authentication token was created as part of the*Prerequisites* on page 648 in the PingIntelligence section. Following is a snippet the showing certificate location and sideband authentication token:
8. **Configure ASE request and response:** Configure ASE request and response API endpoints in `nginx.conf`. Following snippet of `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
```

9. **Apply PingIntelligence policy**: You can apply PingIntelligence policy at the global level, that is, for all the APIs in your environment or for an individual API.
   - **Apply PingIntelligence policy globally**: To apply PingIntelligence policy globally, add `ase_integration_request` and `ase_integration_response` in the `server`section of `nginx.conf` as shown below:

```
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
```

   - **Apply PingIntelligence policy for a specific API**: Apply PingIntelligence modules for APIs by configuring `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 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
```

10. **Verify syntactical correctness of nginx.conf**: 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
```

Final configuration steps: Complete the following steps to configure 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.

```
# 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
```

Following is a sample `nginx.conf` file:

```
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")';
    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 ~* ([^\s]+)(\s+)([^\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;
       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;
    }
}

}
```

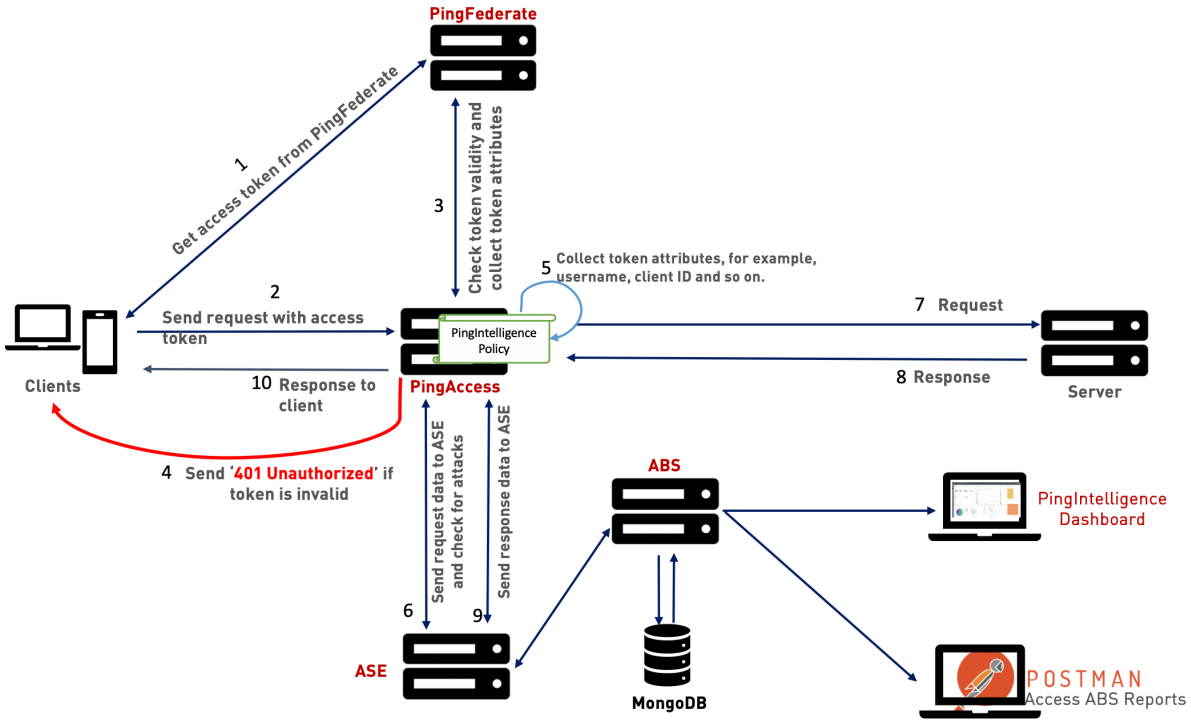
PingAccess API gateway integration

PingAccess sideband integration

This guide describes the deployment of PingIntelligence for APIs in a sideband configuration with PingAccess. A PingIntelligence 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 4.2 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.



Here is the traffic flow through the PingAccess and PingIntelligence for APIs components.

1. Client requests and receives an access token from PingFederate.
2. 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 AI generated Blacklist. 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.
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.

Prerequisites

Complete the following before configuring PingAccess:

- **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.

- **Install PingIntelligence software**

  PingIntelligence software should be installed and configured. For more information on PingIntelligence deployment, see *PingIntelligence for APIs setup* on page 42 and *PingIntelligence manual deployment* on page 75 .
- **Verify that ASE is in sideband mode**

  Check 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)
```

  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.
- **Enable sideband authentication**

  For secure communication between PingAccess and ASE, enable sideband authentication by entering the following ASE command.

```
# ./bin/cli.sh enable_sideband_authentication -u admin –p
```

- **Generate sideband authentication token**

  A token is required for PingAccess to authenticate with ASE. To generate the token, enter the following ASE command.

```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

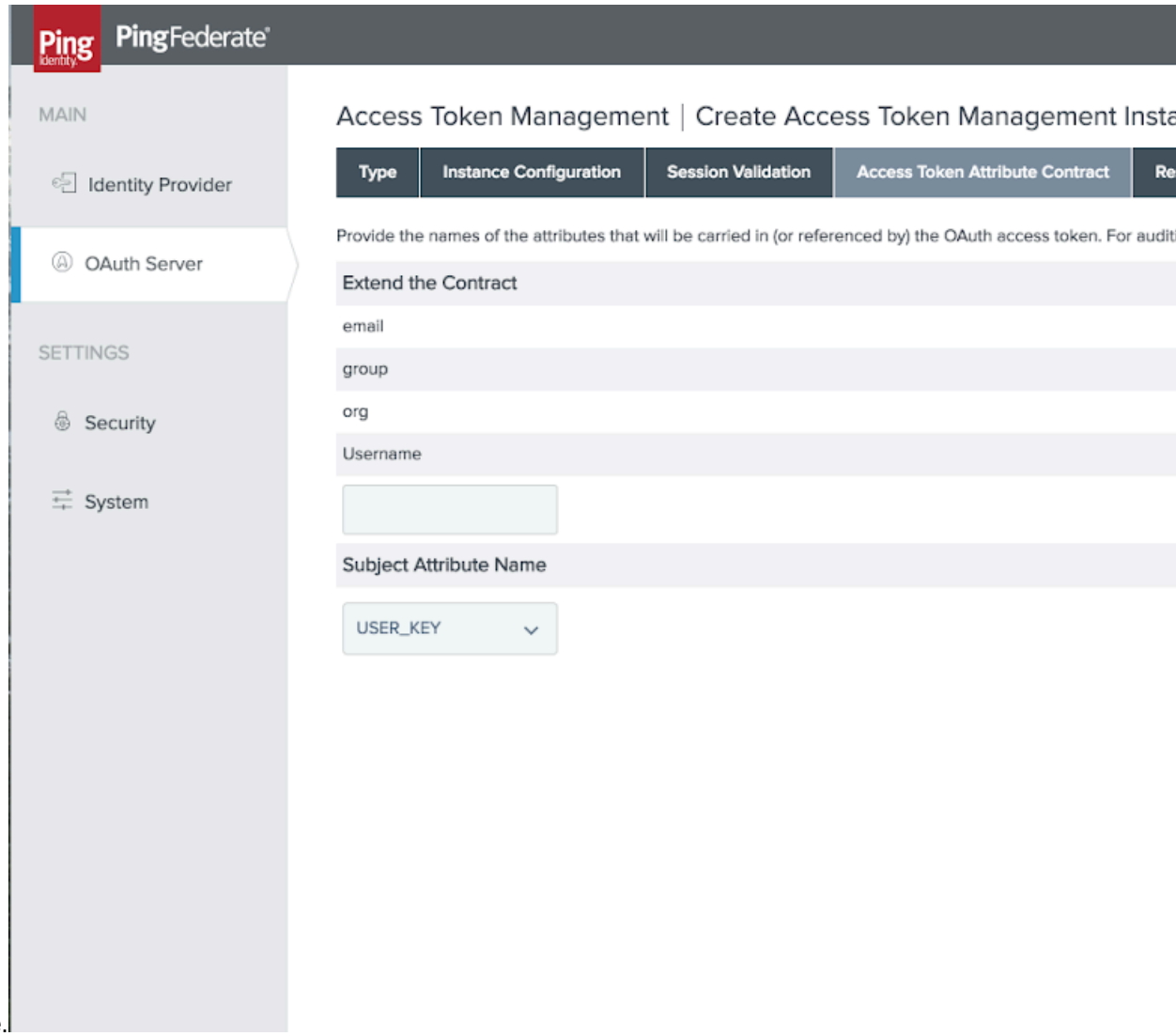
  Save the generated authentication token for further use.

Configure PingFederate to extract token attributes

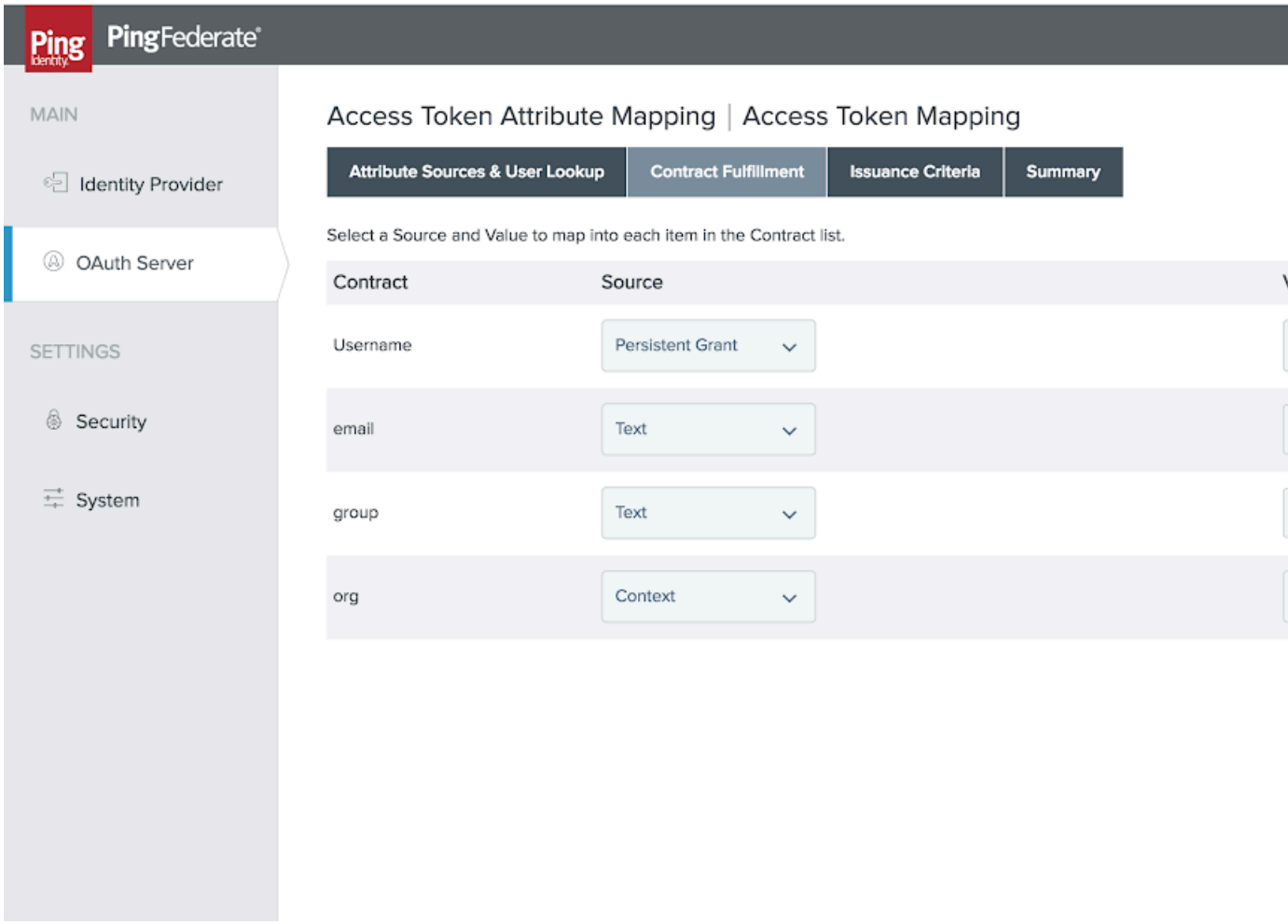
You need to configure PingFederate for PingIntelligence policy to be able to extract the username from the incoming token. Complete the following steps to configure PingFederate to extract token attributes:

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 **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:
   a. Click **Access Token Mapping**
   b. Select the relevant **Context**
   c. Click **Contract Fulfilment**



Deploy the PingIntelligence policy

About this task

To integrate PingAccess with PingIntelligence components, complete the following steps in PingAccess:

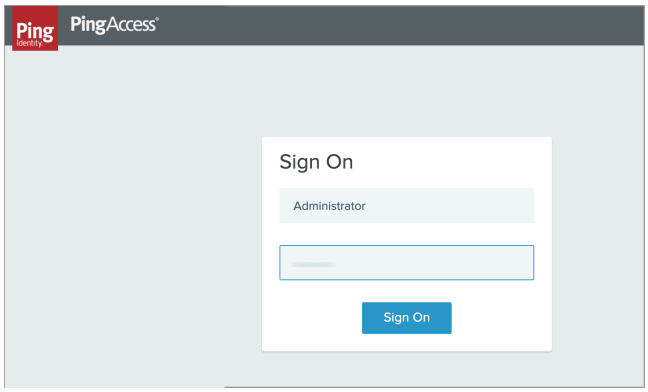
ⓘ **Note:** We recommend that you increase the default heap size in PingAccess before deploying the PingIntelligence policy for PingAccess 6.x. Refer to the instructions explained in *Modifying the Java heap size* for changing the default heap size. For more information, contact PingIdentity support.

Steps

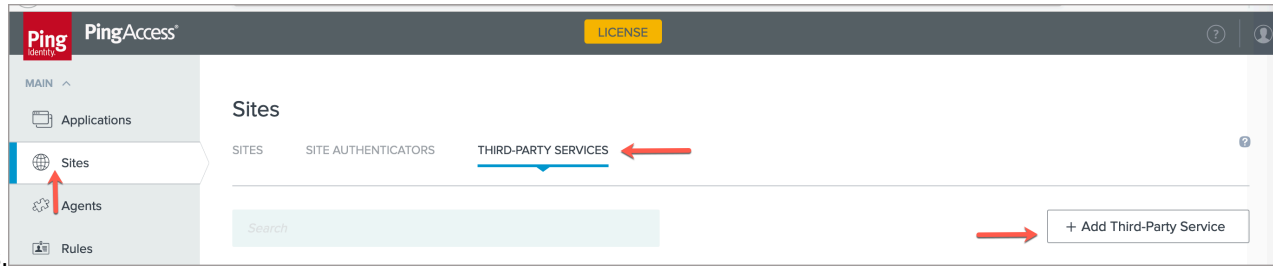
1. Download the PingIntelligence policy from the *Ping Identity download site* and unzip it. The zip file contains three policy files based on the JDK version. Use the policy based on your deployment environment.

```
pingidentity /
└──pingaccess-policy
    ├──openjdk11
    │    └──  pingIntelligence.jar
    ├──oraclejdk11
    │    └──  pingIntelligence.jar
    ├──oraclejdk8
    │    └──  pingIntelligence.jar
    └──version.txt
```

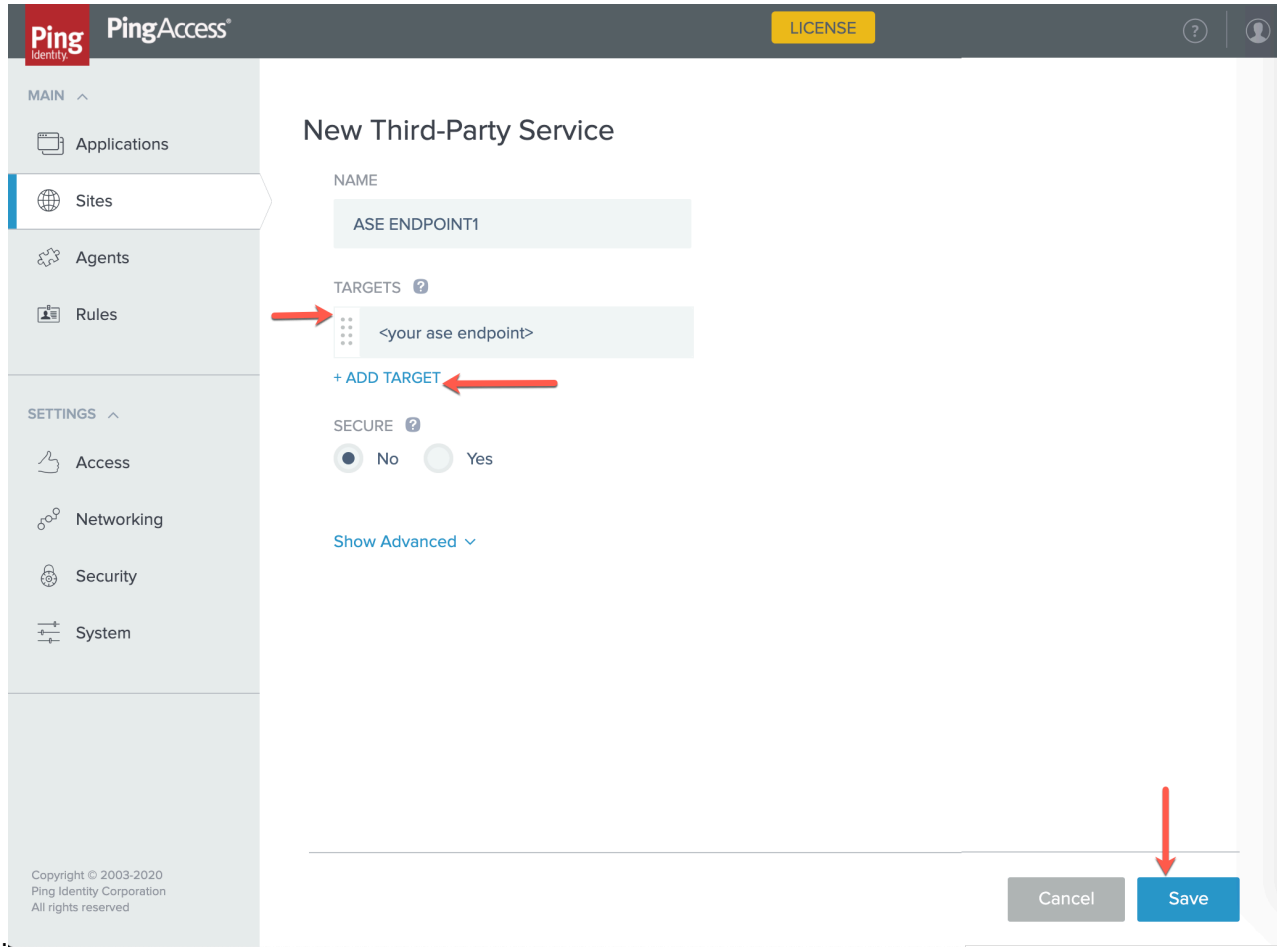
2. Copy the `PingIntelligence.jar` file into the `lib` directory in `PA_home`.
3. Restart PingAccess.
4. Log in to PingAccess.



5. Add the Primary and Secondary ASEs :
   a. In the left pane click **Sites**. Navigate to **THIRD-PARTY SERVICES** and click **+ Add Third-Party Service** to add the two ASEs.



   b. In the **New Third-Party Service** page add the Primary and Secondary ASE endpoints and click **Save**.



> ⓘ **Note:** Select options under **SECURE** to connect PingAccess to PingIntelligence ASE using HTTPS.

6. Add a Rule for the Primary and Secondary ASEs :
   a. In the left pane click **Rules**. In the new Rule page, in the **NAME** field, enter the name of the rule for PingIntelligence.
   b. In the **TYPE** drop-down list, select **PingIntelligence**. This appears in the drop-down list after adding PingIntelligence.jar in PA_HOME in step 3.
   c. Select the ASE Endpoint for Primary ASE in **PINGINTELLIGENCE ASE ENDPOINT** drop-down list.
   d. Select the ASE Endpoint for Secondary ASE in **PINGINTELLIGENCE 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.

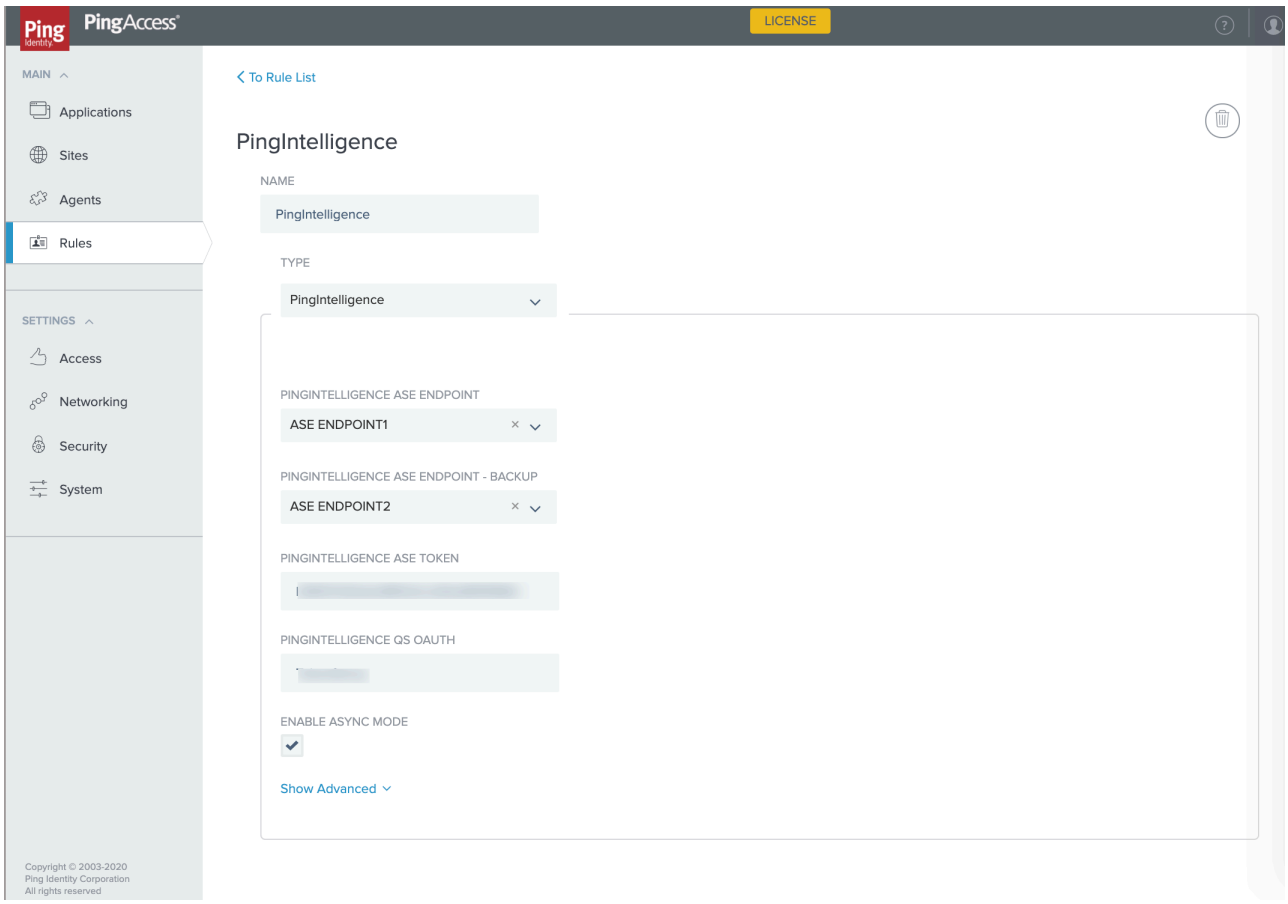
   e. In the **PINGINTELLIGENCE ASE TOKEN** field, enter ASE sideband token that is generated for authentication between PingAccess and ASE.
   f. If an OAuth token comes as part of a query string, enter the name of the query string in the **PINGINTELLIGENCE QS OAUTH** field.

   ⓘ **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.

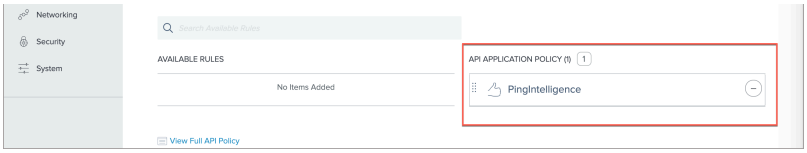
   g. Select the **ENABLE ASYNC MODE** to choose Asynchronous mode between PingAccess and ASE.

   ⓘ **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

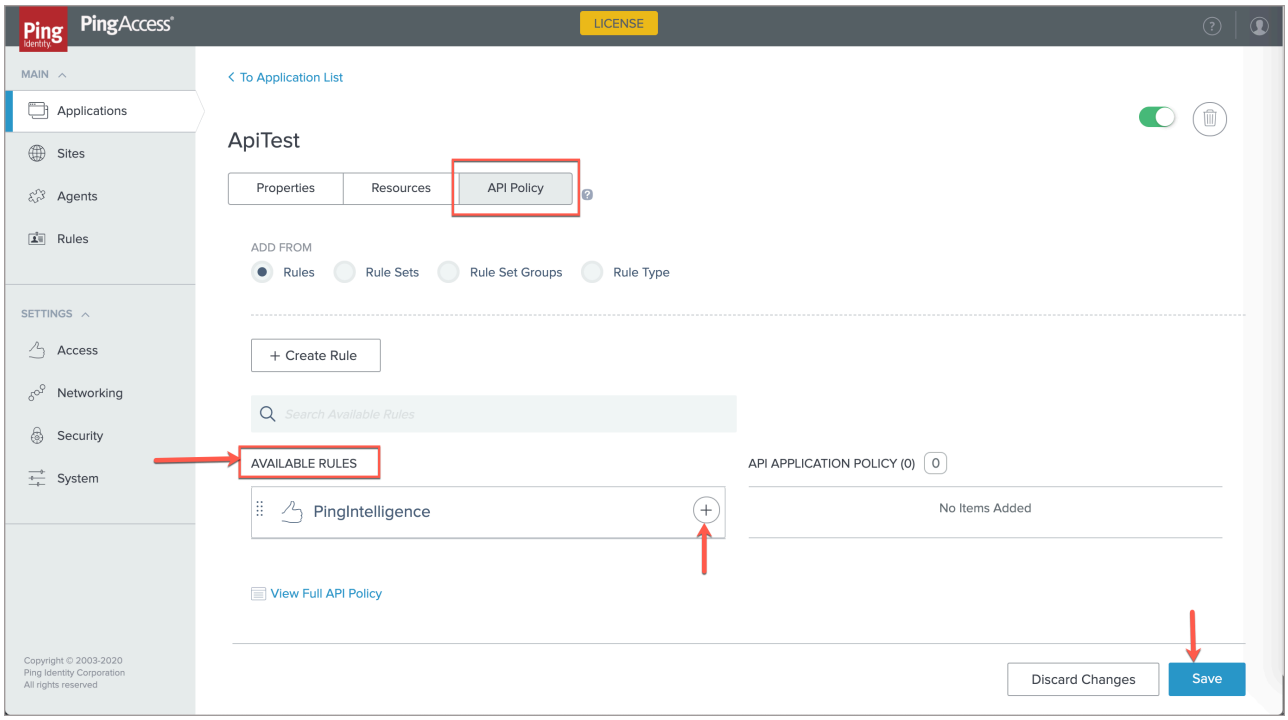
from ASE and sends the request to backend server. ASE performs attack detection without blocking of attacks in asynchronous mode.



7. Apply the rule by completing the following steps :
   a. Edit the existing application.
   b. In the edit application page, click on **API Policy**.
   c. Under **Available Rules,** Click the ⊕ sign for the PingIntelligence rule.
   d. After clicking on the ⊕ sign, the PingIntelligence rule moves under the **API APPLICATION POLICY** as shown in the screen capture below.



   e. Click **Save** to save the rule.



Configure ASE persistent connection

You can optionally configure TCP keep-alive connections in the `ase.conf` file of ASE. Following is a snippet of `ase.conf` displaying the `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
```

API discovery

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 JSON is defined in the ASE. For more information, see *API discovery and configuration* on page 321 . You can edit the discovered API's JSON definition in Dashboard before adding them to ASE. For more information on editing and configuring API discovery, see *Discovered APIs* on page 471.

Configure API discovery in PingAccess

To configure API discovery in PingAccess add Applications in PingAccess with **Application Type** as API. For PingIntelligence Dashboard to automatically capture the definition, include the following parameters in the **DESCRIPTION** section when you add an Application:

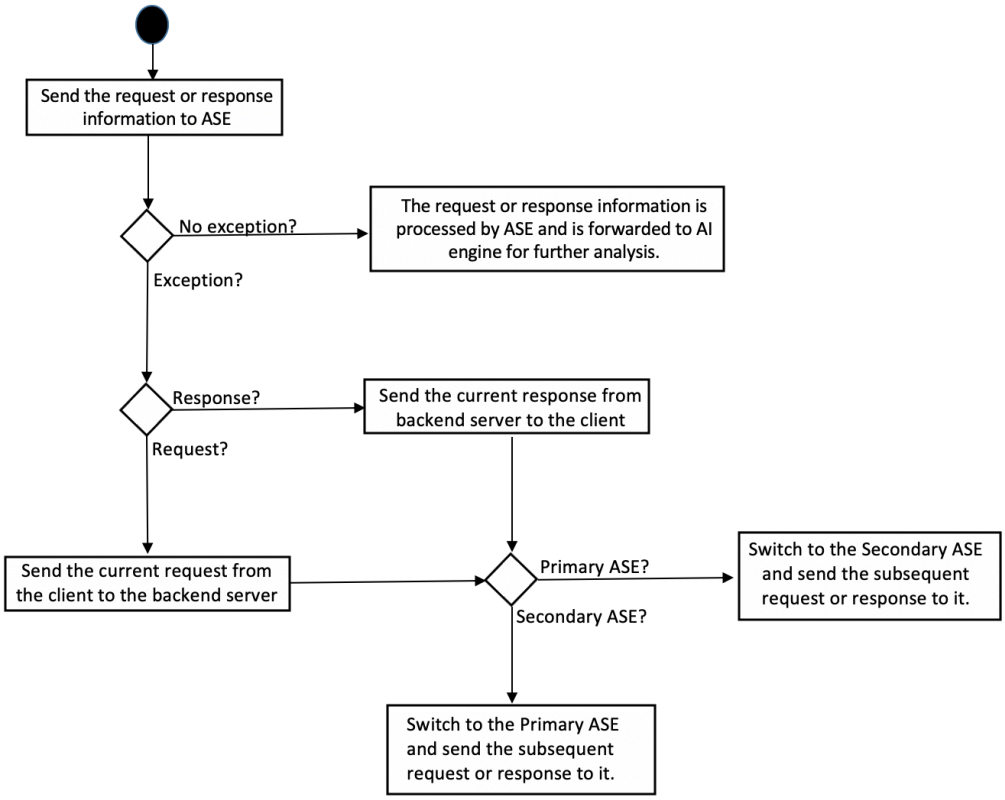
```
{
"ping_ai": true,
"ping_host": "",
"ping_url": "",
"ping_login": "",
"ping_cookie": "JSESSIONIDTEST",
"apikey_qs": "X-API-KEY",
"apikey_header": "",
"ping_decoy": false,
"oauth2_access_token": false,
"ping_blocking": true
}
```

The following table describes the parameters captured when PingIntelligence Dashboard fetches the API definition from PingAccess and adds it to ASE.

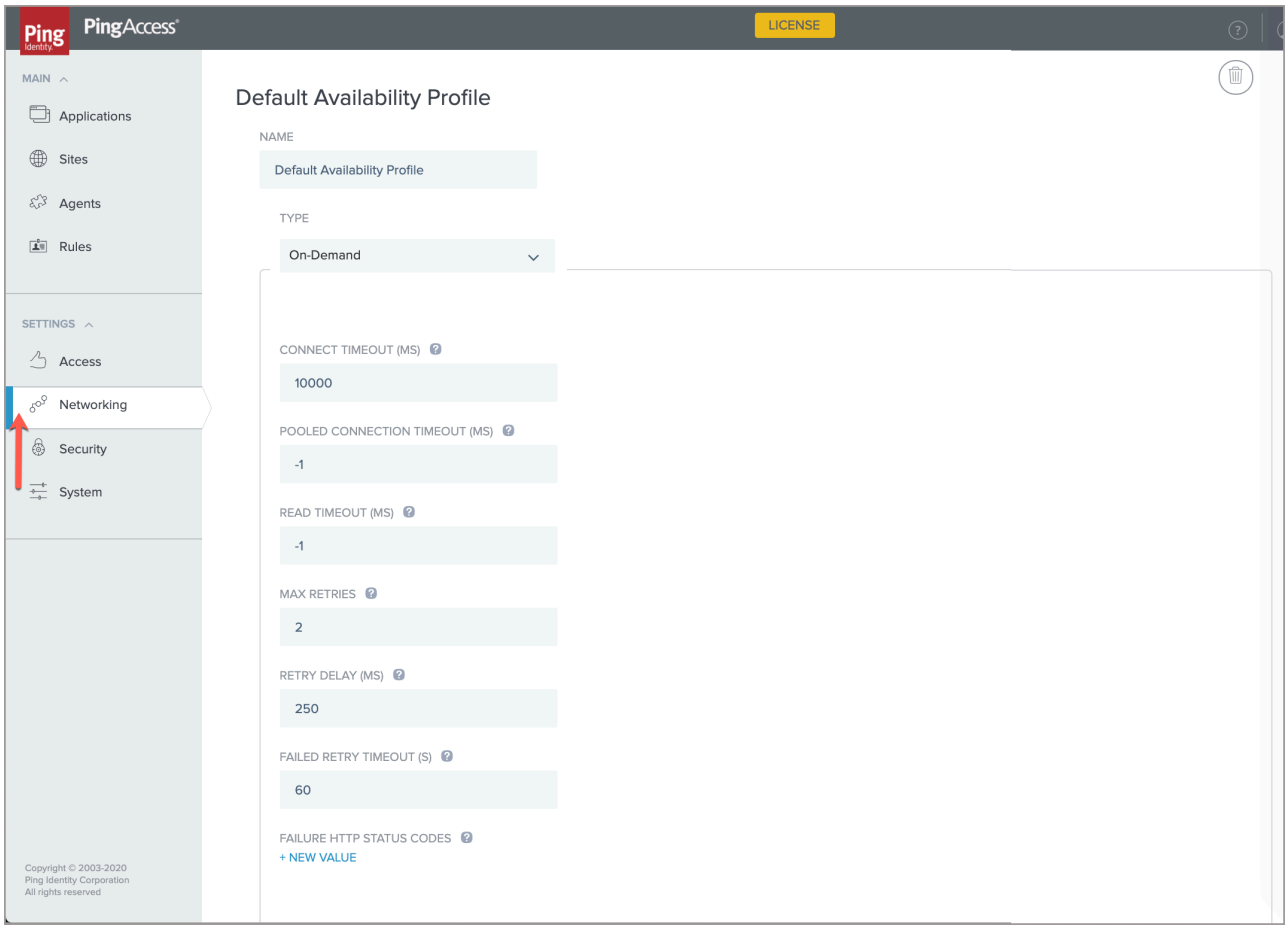
Parameter	Description
`ping_ai`	When `true`, PingIntelligence processing is applied to this API. Set to `false` for no PingIntelligence processing. Default value is `true`.
`ping_host`	Hostname of the API. You can configure `*` as `hostname` to support any hostname.
`ping_url`	The base URL of the managed API, for example, `/shopping`. This field cannot be empty.
`ping_login`	Login 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.
`apikey_header`	When API Key is part of the header field, ASE uses the specified parameter name to capture the API key value.
`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`.

Handle exceptions

This topic discusses exception handling by 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.



You can configure an **Availability profile** to define the way PingAccess manages network requests. For more information, see *Availability profiles*



WSO2 API gateway integration

PingIntelligence 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 WSO2 API Gateway is the primary component that intercepts API requests and applies various types of policies. Each policy is executed using something we call 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 Ping 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 as well. With the provided request metadata Ping ASE can detect abnormal access patterns. It also builds 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*.

PingIntelligence Docker toolkit

PingIntelligence Docker toolkit

PingIntelligence for APIs provides a Docker toolkit which is used to create Docker images of PingIntelligence components and MongoDB. The Docker toolkit can be run on either on a RHEL 7.6 or a Ubuntu 16.0.4 LTS machine. 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 PoC deployment* on page 28

Prerequisites: *Download* the following PingIntelligence components, tools, and open source modules.

- **Download products:**
  - PingIntelligence ASE 4.2
  - PingIntelligence ABS 4.2
  - PingIntelligence Dashboard 4.2
  - MongoDB 4.2.0
  - OpenJDK 11.0.2 to 11.0.6
  - Kibana 6.8.1
  - Elasticsearch 6.8.1
- **License** Obtain valid ASE and ABS license files from the PingIntelligence sales team.

> ⓘ **Note:**
>
> - Make sure to download the correct ASE binary (RHEL7.6 or Ubuntu 16 LTS) based on the base image you want to create.
> - Download the correct MongoDB 4.2.0 binary (RHEL7.6 or Ubuntu) based on the Docker image you want to build.

Untar the Docker toolkit

To use the Docker toolkit, you need to untar the toolkit. Run the following command to untar the toolkit:

```
tar -zxf pi-api-docker-toolkit-4.2.tar.gz
```

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
`data`	For internal use

`external`	Contains the third-party software: ▪ MongoDB 4.2.0 ▪ Elasticsearch 6.8.1 ▪ Kibana 6.8.1 ▪ OpenJDK 11.0.2 to 11.0.6
`images`	Contains the created Docker images using the `build.sh` script
`lib`	For internal use
`license`	Contains the `ase` and `abs` directory to copy the respective license files. ⓘ **Note:** You can build the images without adding the license files to the `ase` and `abs` directory. If you build the Docker images without the license file in `ase` and `abs` directory, then you need to map or mount the license file in the following exact location: ▪ ASE: `/opt/pingidentity/ase/config/PingIntelligence.lic` ▪ ABS: `/opt/pingidentity/abs/config/PingIntelligence.lic`
`logs`	Contains the log files
`software`	Contains PingIntelligence ASE, ABS, and Dashboard

Configure docker.conf

Navigate to the `config` directory and edit the `docker.conf` file for base image name and base image operating system. Following is a sample `docker.conf` field:

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

# 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
```

ⓘ **Note:** Do not set the `user_name` as `root` in `docker.conf` file.

Build the PingIntelligence Docker images

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

- ASE
- ABS
- Dashboard
- MongoDB

It is a good practice to obfuscate the various keys and password in ASE, ABS, and Dashboard before building the Docker images. For more information on obfuscating keys and passwords, see the following topics:

- ASE - *Obfuscate keys and passwords*
- ABS - *Obfuscate passwords*
- Dashboard - *Obfuscate keys and passwords*

Complete the following steps to build the Docker images:

1. Configure the base image name and base image operating system details in the `config/docker.conf` file.
2. Download the PingIntelligence software in the `software` directory and save them with the name as shown in the following table:

Software	**File name**
ASE	`pi-api-ase-rhel-4.2tar.gz`
ABS	`pi-api-abs-4.2.tar.gz`
Dashboard	`pi-api-dashboard-4.2.tar.gz`

3. Download OpenJDK 11.0.2, Kibana 6.8.1, Elasticsearch 6.8.1 and MongoDB 4.2.0 in the `external` directory and save them with the name as shown in the following table:

Software	**File name**
Elasticsearch	`elasticsearch.tar.gz`
OpenJDK 11.0.2	`openjdk11.tar.gz`
Kibana	`kibana.tar.gz`
MongoDB	`mongodb.tgz`

ⓘ **Note:** Make sure that MongoDB is as per the base image configured in `docker.conf` file.

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:4.2
Image saved to /home/ubuntu/docker-setup/images/pingidentity_ase.tar
Creating build context for abs
Creating Image
Image created with tag pingidentity/abs:4.2
Image saved to /home/ubuntu/docker-setup/images/pingidentity_abs.tar
Creating build context for dashboard
Creating Image
Image created with tag pingidentity/dashboard:4.2
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
```

The other options that you can give with `build.sh` are: `ase`, `abs`, `dashboard`, and `mongo`.

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

```
sudo docker image ls | grep pingidentity
pingidentity/dashboard              4.2                 4c2b1378bec0
 38 minutes ago      2.07GB
pingidentity/mongo                  4.2.0               fcf3986eab34
 3 days ago          781MB
pingidentity/abs                    4.2                 feced8848a28
 3 days ago          910MB
pingidentity/ase                    4.2                 5ab75d3180ed
 3 days ago          595MB
```

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

```
docker-setup# ls -ltra images/
total 3437116
drwxr-xr-x 11 root root       4096 Mar 18 18:39 ..
-rw-------  1 root root  782182400 Mar 21 10:18 pingidentity_abs.tar
-rw-------  1 root root  495038976 Mar 21 10:20 pingidentity_mongo.tar
-rw-------  1 root root  339437568 Mar 23 06:57 pingidentity_ase.tar
-rw-------  1 root root 1302484480 Mar 23 08:08 pingidentity_dashboard.tar
drwxr-xr-x  2 root root       4096 Mar 23 08:08 .
```

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

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 ASE, 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*.
`ASE_TIMEZONE`	`string`	Set the timezone of ASE to either `local` or `UTC`. Default value is `local`.
`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:** 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. Once enabled, ASE authenticates requests using ASE authentication tokens.
ENABLE_GOOGLE_PUBSUB	true/false	Google Pub/Sub configuration
GOOGLE_PUBSUB_TOPIC	string	
GOOGLE_PUBSUB_CONCURRENCY	number	Number of concurrent connections to Google Pub/Sub Minimum: 1, Default: 1000, Maximum: 1024
GOOGLE_PUBSUB_QPS	number	Number of messages published per second. Minimum: 1, Default: 1000, Maximum: 10000
GOOGLE_PUBSUB_APIKEY	string	Google service account API key (Optional)
CACHE_QUEUE_SIZE	number	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
GOOGLE_PUBSUB_TIMEOUT	number	Timeout in seconds to publish a message to Google Pub/Sub. Minimum: 10, Default: 30, Maximum: 300

ABS Environment Variables: The following table lists the ABS environment variables and the values:

Environment	Value	Usage
MONGO_RS	<IP or hostname>:port	MongoDB replica set IP address or host name and port.
MONGO_USERNAME	string	MongoDB username
MONGO_PASSWORD	string	MongoDB password
HOST_IP	IP or hostname	If you have multiple network interfaces or if you are running inside a Docker, specify the externally visible IP address for ABS to bind
ABS_LOG_LEVEL	string	Log levels (ALL > DEBUG > INFO > WARN > ERROR > FATAL > OFF) 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. 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	Email address used for sending email alerts and reports.
SENDER_EMAIL_PASSWORD	string	Password of sender's email account. ⓘ **Note:** You can leave this field blank if your SMTP server does not require authentication.
RECEIVER_EMAIL	string	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 CLI admin password.

MongoDB Environment Variables: The following table lists the MongoDB environment variables and the values:

Environment	Value	Usage
POC_MODE	string	Sets the mode in which ABS trains its API models. Set it to `true` for running ABS in POC mode. For more information, see*ABS POC mode* on page 284
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_ACCESS_KEY_RU	string	The access key for the restricted user. For more information on restricted user, see *ABS users*.
ABS_SECRET_KEY_RU	string	The secret key for the restrict ired user. For more information on restricted user, see *ABS users*.
MONGO_USERNAME	string	MongoDB username
MONGO_PASSWORD	string	MongoDB password
MUTLI_NODE_REPLICA_SET	true/false	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.
ATTACK_INITIAL_TRAINING	integer	The attack training period
ATTACK_UPDATE_INTERVAL	integer	Attack threshold uphold interval
API_DISCOVERY	true/false	Set the value to true to enable API discovery in ABS. For ABS to discover APIs, a global API JSON must be configured in ASE. See *API discovery* for more information.
API_DISCOVERY_INITIAL_PERIOD	integer	The initial period set in hours in which ABS has to be discover APIs. It is a 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.
WIRED_TIGER_CACHE_SIZE_GB	float	Memory in GB to be used by MongoDB cache.
MONGO_SSL	string	Configures whether MongoDB uses SSL. Default values is false.

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	URL of PingAccess if you set the discovery source as `pingaccess`.
PINGACCESS_USERNAME	string	PingAccess username for API discovery.
PINGACCESS_PASSWORD	string	PingAccess password for API discovery.
AXWAY_URL	string	URL of Axway if you set the discovery source as `axway`.
AXWAY_USERNAME	string	Axway username for API discovery.
AXWAY_PASSWORD	string	Axway username for API discovery.
DISCOVERY_MODE	string	Mode in which Dashboard publishes APIs to ASE. Values can be `auto` or `manual`. For more information, see*Discovered APIs* on page 471
DISCOVERY_MODE_AUTO_POLLING_INTERVAL	integer	If the `DISCOVERY_MODE` is set as `auto`, set the polling interval at which Dashboard polls the discovery source for APIs. It is recommended to have minimum value of 10-minutes.
DISCOVERY_MODE_AUTO_DELETE_MODE	string	If the `DISCOVERED_APIS` is set as `auto`, you can choose to retain to delete APIs in ASE which 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 that 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	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 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 ABS restricted user. For more information on restricted user, see *ABS users*.
ABS_URL	string	The URL should be in the form of `https://<IP>:<port>`. The URL is used by Web GUI to connect to ABS.
ASE_URL	string	The URL should be in the form of `https://<IP>:<port>`. The URL is used by Web GUI to connect to ASE.
ASE_ACCESS_KEY	string	Access key of the ASE admin user
ASE_SECRET_KEY	string	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 Web GUI to connect to dashboard. IP and port number are of Kibana.
H2_DB_PASSWORD	string	Password for H2 database
H2_DB_ENCRYPTION_PASSWORD	string	Password to change encryption method of H2 database
WEBGUI_ADMIN_PASSWORD	string	Password for admin user of Web GUI
WEBGUI_PING_USER_PASSWORD	string	Password for `ping_user` of Web GUI

Using environment variables - example

The following sections show example of using environment variables to create containers. The containers must be created in the following order:

1. MongoDB
2. ABS
3. ASE
4. Dashboard

Launch MongoDB container: Run the following command with some sample environment variables to launch the MongoDB container:

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

Running this command creates the MongoDB container with settings in environment variable provided. If any of the environment variable is not used, then the container is launched with default values.

Launch ABS container: Run the following command with some sample environment variables to launch the ABS container:

```
docker run -d --name abs --hostname abs --link mongo:mongo -e
 MONGO_RS=mongo:27017 \
-e MONGO_USERNAME="new_mongo_user" \
-e MONGO_PASSWORD="new_mongo_password" -e MONGO_SSL="true" \
pingidentity/abs:4.2
```

Launch 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 -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" --shm-size=1g
 pingidentity/ase:4.2
```

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 \
-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:4.2
```

Launch Dashboard: Run the following command with some sample environment variables to launch the Dashboard container:

```
docker run -d --name webgui --link abs:abs --link ase:ase --hostname webgui
 \
-e ABS_RESTRICTED_USE_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 DASHBOARD_URL="https://localhost:5601" -e
 WEBGUI_ADMIN_PASSWORD="new_webgui_admin_password" \
-e WEBGUI_PING_USER_PASSWORD="new_webgui_pinguser_password" \
-e WEBGUI_ADMIN_PASSWORD="new_webgui_admin_password" \
-p 8030:8030 pingidentity/dashboard:4.2
```

Port mapping

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

	9090	Access log upload port number
Dashboard	8030	Dashboard port number
MongoDB	27017	MongoDB port number

PingIntelligence Hardening Guide

PingIntelligence security hardening guide

The PingIntelligence for APIs hardening document provides administrators with a single point of reference for configurations and best practices available to harden their PingIntelligence for APIs platform. To avoid duplication, this document does to include the detailed configuration instructions. Instead, it refers readers to information on hardening the PingIntelligence platform.

Ping security overview

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 document 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 administrators work on deploying a component in PingIntelligence, they can look up the corresponding section in this document.

Accessing the document

Click on *PingIntelligence security hardening guide*, to access this document. You must have a registered account to the *PingIdentity support and community portal*, for accessing it. If you need any further assistance, contact Ping sales team.