

# PingIntelligence



# Contents

<b>PingIntelligence for APIs Overview.....</b>	<b>8</b>
<b>PingIntelligence for APIs Release Notes.....</b>	<b>8</b>
PingIntelligence 4.0.2 Release Notes.....	8
PingIntelligence 4.0.1 Release Notes.....	9
PingIntelligence 4.0 Release Notes.....	10
<b>API Security Enforcer.....</b>	<b>11</b>
Introduction.....	11
Administration.....	13
ASE license.....	13
ASE interfaces.....	13
Customizing ASE ports.....	15
Tune host system for high performance.....	16
Start and stop ASE.....	16
Change default settings.....	17
Obfuscate keys and passwords.....	18
PKCS#12 keystore.....	19
Directory structure.....	19
ASE cluster setup.....	20
Configure SSL for external APIs.....	25
Configure SSL for management APIs.....	28
Configure native and PAM authentication.....	30
ASE management, access and audit logs.....	32
Purge log files.....	34
Configure syslog.....	35
Email alerts and reports.....	35
Sideband ASE.....	42
ASE configuration - ase.conf.....	44
Activate API cybersecurity.....	54
API deception environment.....	63
ABS AI-based security.....	65
CLI for sideband ASE.....	68
Inline ASE.....	75
ASE configuration - ase.conf.....	76
API routing.....	87
Real-time API cybersecurity.....	90
API deception environment.....	103
ASE DoS and DDoS protection.....	106
ABS AI-based security.....	116
CLI for inline ASE.....	119
ASE REST APIs using Postman.....	128
ASE self-signed certificate with Postman.....	129
View ASE REST APIs in Postman.....	129
REST API for inline and sideband ASE.....	132
Audit log.....	155
Supported encryption protocols.....	158

Autoscaling ASE in AWS environment.....	159
Create an AMI for ASE.....	159
Create an IAM role in the security, identity, and compliance.....	160
Create the security group.....	162
Create launch configuration.....	162
Create an auto-scale group.....	163
ASE log messages.....	163

## **ABS AI Engine..... 165**

Features at a glance.....	165
Introduction.....	166
Administration.....	166
ABS License.....	167
Start and Stop ABS.....	167
Change default settings.....	168
ABS users for API reports.....	170
ABS directory structure.....	170
Obfuscate passwords.....	171
Configure SSL.....	172
Import existing CA-signed certificates.....	173
ABS ports.....	173
ABS configuration - abs.properties.....	174
Connect ABS to API Security Enforcer.....	177
ABS cluster.....	180
ABS logs.....	181
Purge the processed access logs from ABS.....	182
Purge MongoDB data.....	182
Reset MongoDB.....	183
Email alerts and reports.....	187
ABS REST API format.....	191
Admin REST API.....	192
AI Engine training.....	195
Training the ABS model.....	195
AI Engine training variables.....	195
Training period status.....	197
Update the training variables.....	197
Tune thresholds for false positives.....	199
Disable attack detection.....	203
API discovery.....	203
Reporting the discovered APIs.....	204
Discovery Subpaths.....	205
Root API in ASE and attack detection.....	206
ABS Discovery API.....	207
Manage discover intervals.....	210
Automated API Definition (AAD).....	210
Install AAD.....	213
Obfuscate keys and passwords.....	213
AAD deployment options.....	214
Configure AAD.....	216
Start and stop AAD.....	220
Start AAD in offline mode.....	220
AAD Conversion Status.....	221
Purge AAD log files.....	224
REST API attacks.....	224
REST API attack types.....	225

Attacks based on username activity.....	227
Attacks based on API Key activity.....	228
Attacks based on cookie activity.....	228
Attacks based on token activity.....	230
Attacks based on IP activity.....	231
WebSocket API attack detection.....	232
Manage Attack blocking.....	234
Automated attack blocking.....	234
Whitelist and blacklist management.....	234
TTL for client identifiers.....	237
ABS blacklist management and reporting.....	240
Per API blocking.....	242
Attack Management Tool.....	243
Unblock client.....	245
Tune thresholds.....	245
Attack reporting.....	246
Consolidated result of attack types.....	246
API Deception.....	250
Real-time Detected attacks for inline ASE.....	252
Anomalous activity reporting.....	255
Blocked connection reporting.....	256
API forensics reporting.....	258
API metrics reporting.....	263
Username based metrics.....	266
API Key based metrics.....	268
OAuth token based metrics.....	271
List valid URL.....	274
Hacker's URL.....	275
Backend error reporting.....	277
API DoS and DDoS threshold.....	277
API reports using Postman.....	279
ABS self-signed certificate with Postman.....	279
View ABS reports in Postman.....	280
ABS and AAD CLI.....	283
ABS external REST APIs.....	285
Admin REST API.....	285
Discovery REST API.....	288
Decoy REST API.....	289
GET Threshold REST API.....	291
PUT Threshold REST API.....	294
Metrics REST API.....	295
API Key Metrics REST API.....	298
OAuth2 Token Metrics REST API.....	300
Username Metrics REST API.....	303
Anomalies REST API.....	306
Anomalies across APIs.....	308
OAuth2 Token Forensics REST API.....	309
IP Forensics REST API.....	311
Cookie Forensics REST API.....	313
Token Forensics REST API.....	314
API Key Forensics REST API.....	315
Username Forensics REST API.....	317
Attack Types REST and WebSocket APIs.....	318
Flow Control REST API.....	319
Blocked Connection REST API.....	320
Backend Error REST API.....	321

List Valid URLs REST API.....	323
List Hacker's URL REST API.....	325
Threshold range for Tn and Tx.....	326
Splunk for PingIntelligence.....	330
System requirements.....	331
Install and configure Splunk for PingIntelligence.....	331
Types of data captured.....	334
Local Input Method installation and configuration.....	339
Splunk Universal Forwarder method installation and configuration.....	345
Alert notification on Slack and Email.....	352
ABS log messages.....	356
AAD log messages.....	359

## **PingIntelligence Dashboard.....360**

PingIntelligence for APIs Dashboard.....	360
System requirements for Dashboard.....	361
Dashboard installation.....	361
Prerequisites.....	362
Install PingIntelligence Dashboard software.....	362
Download and install Elasticsearch.....	363
Install Kibana.....	365
Install Ping styling plugin.....	366
Configure Dashboard.....	366
Update the admin password.....	368
Obfuscate keys and passwords.....	369
Start Dashboard.....	370
Dashboard fast forward.....	370
Syslog for attack data.....	372
Access the PingIntelligence Dashboard.....	373
View blacklist.....	383
Dashboard time series.....	387
Stop Dashboard.....	389
Purge data from Elasticsearch.....	389
Purge dashboard logs.....	389
Dashboard log messages.....	390

## **PingIntelligence Deployment.....391**

Automated deployment.....	391
Automated PingIntelligence for APIs setup.....	391
ASE deployment modes.....	393
Step 1: Setup host system.....	395
Change default settings.....	402
Step 2: Configure system parameters.....	410
Step 3: Install the PingIntelligence for APIs software.....	411
Install PingIntelligence as a systemd service.....	412
Verify PingIntelligence Installation.....	413
Next steps - Integrate PingIntelligence into your environment.....	415
Shut down the deployment.....	416
Logs.....	417
Manual deployment.....	417
PingIntelligence manual deployment.....	417
Part A – Install ASE.....	417
Part B – Install ABS and MongoDB.....	430
Part C – Integrate ASE and ABS.....	441

Part D – Install PingIntelligence Dashboard software.....	443
Part E – Access ABS reporting.....	455
Part F - Integrate API gateways for sideband deployment.....	460

## **API Gateway Integration..... 460**

Apigee API gateway integration.....	460
PingIntelligence Apigee Integration.....	460
Prerequisites to deploying PingIntelligence shared flow.....	461
Download and configure automated policy tool.....	462
Deploy the PingIntelligence policy.....	464
Change deployed policy mode.....	474
Add APIs to ASE.....	476
AWS API gateway integration.....	476
PingIntelligence AWS API Gateway Integration.....	476
Prerequisites.....	478
Configure automated policy tool.....	481
Deploy PingIntelligence Policy for AWS.....	491
Next steps - Integrate into your API environment.....	491
API discovery.....	492
Uninstall CloudFront sideband policy.....	495
Axway API gateway integration.....	498
Axway sideband integration.....	498
Azure API gateway integration.....	525
Azure APIM sideband integration.....	525
Prerequisites.....	527
Deploy PingIntelligence policy.....	529
Integrate PingIntelligence.....	531
Configure ASE persistent connection.....	532
API discovery.....	532
CA API gateway integration.....	536
PingIntelligence - CA API gateway sideband integration.....	536
Prerequisite.....	537
Install and configure the PingIntelligence bundle.....	538
Import PingIntelligence policy.....	541
Configure ASE token and certificate.....	542
Apply PingIntelligence policy.....	542
Integrate PingIntelligence.....	544
API discovery.....	544
PingAccess API gateway integration.....	548
PingAccess sideband integration.....	548
Mulesoft API gateway integration.....	561
Mulesoft sideband integration.....	561
Prerequisites.....	562
Deploy PingIntelligence policy.....	563
Apply PingIntelligence policy.....	570
Next steps - Integration.....	576
API discovery.....	576
NGINX integration.....	580
NGINX sideband integration.....	580
Prerequisites.....	582
NGINX for RHEL 7.6.....	584
NGINX for Ubuntu 16.04.....	590
Next steps - integration.....	595
API discovery.....	596
WSO2 API gateway integration.....	599

PingIntelligence WSO2 integration.....	599
<b>PingIntelligence PoC.....</b>	<b>600</b>
Docker - inline PoC deployment.....	600
Docker Inline PoC setup.....	600
Installation requirements.....	602
Download and untar Docker package.....	602
Install and load Docker images.....	604
Setup the PoC environment.....	604
Start the training.....	605
Generate sample attacks.....	605
API deception.....	605
API discovery.....	606
View dashboard reports.....	607
ABS detailed reporting.....	609
Shutdown the PoC environment.....	612
Appendix: Verify the Setup.....	612
Cloud PoC service deployment.....	612
PingIntelligence Cloud PoC Service.....	612
Download and install ASE software.....	614
Obfuscate access and secret key.....	617
Configure ASE and Dashboard.....	618
Configure PingIntelligence Cloud Connection.....	618
Start and stop ASE.....	619
Enable ASE to ABS engine communication.....	619
Integrate PingIntelligence into your API environment.....	619
Add APIs to ASE.....	620
Download and install AAD.....	621
AI engine training.....	622
Connect to the PingIntelligence dashboard.....	622
Access ABS reporting.....	624
<b>PingIntelligence Docker toolkit.....</b>	<b>625</b>
PingIntelligence Docker toolkit.....	625
Untar the Docker toolkit.....	625
Build the PingIntelligence Docker images.....	626
Environment variables exposed in Docker images.....	628
Using environment variables - example.....	631
<b>PingIntelligence App Notes.....</b>	<b>633</b>
Dashboard deployment using NGINX.....	633
PingIntelligence Dashboard using Amazon ES.....	634
Dashboard using on-premise Elasticsearch.....	639

# 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 4.0.2 Release Notes

---

New in ABS AI Engine

- **MongoDB:** ABS 4.0.2 now requires MongoDB 4.2.0.
- **MongoDB reset script:** ABS AI Engine provides a MongoDB reset script in the `util` directory. The script does a factory reset of all the MongoDB data. To preserve historical API activity and models, it is recommend to take a backup of MongoDB before running the reset script. For more information, see [Reset MongoDB](#) on page 183.
- **TLS 1.1 and 1.2:** ABS 4.0.2 only support TLS 1.1 and TLS 1.2.

New in ASE

- **Email alert logging:** ASE logs email alerts in `controller.log` even when `email_alert` is disabled in `ase.conf` file. For more information, see **Alerts logged in log file** in [Email alerts and reports](#) on page 35.
- **Inline ASE - Health check headers:** ASE in inline mode now supports optional headers for custom health checks of backend systems. For more information, see [Define an Inline API JSON configuration file](#) on page 83.
- ASE now requires a `nofile` ulimit to be set to a minimum of 65535 for it to start. For more information see, [Start and stop ASE](#) on page 16.

New in PingIntelligence for APIs Dashboard

PingIntelligence 4.0.2 provides the following enhancements:

- Enhance main and API specific dashboard



- Detailed user activity reporting shows API activity for a given user. It includes:
  - APIs accessed and requests per API
  - Total number and type of attacks detected
  - Tokens utilized
  - IP addresses used
  - API methods used

## PingIntelligence 4.0.1 Release Notes

---

### ABS AI engine

#### New in ABS AI engine 4.0.1

- **ABS AI Engine to MongoDB SSL communication:** To enable SSL communication between ABS and MongoDB, configure the new `mongo_ssl` parameter in the `abs.properties` file. For more information, see [ABS configuration - abs.properties](#) on page 174
- **Email communication over SSL:** To enable SSL communication between ABS and a mail server, configure the new parameters, `smtp_ssl` and `smtp_cert_verification` in the `abs.properties` file. For more information, see [ABS configuration - abs.properties](#) on page 174
- **ABS AI Engine EULA acceptance:** ABS `start.sh` script now requires a EULA to be accepted. For more information, see [Start and Stop ABS](#) on page 167.

### ASE

#### New in ASE 4.0.1

- **Email communication over SSL:** To enable SSL communication between ASE and a mail server, configure the new parameters, `smtp_ssl` and `smtp_cert_verification` in the `ase.conf` file. For more information, see [ASE configuration - ase.conf](#) on page 44
- **Parsing statistics summary in balancer log:** ASE `balancer.log` file now provides a summary of sideband processing statistics including error reporting. The log file displays the number of errors reported under various categories. For more information, see balancer logs in [ASE management, access and audit logs](#) on page 32
- **Postman collection:** A new Postman collection simplifies use of ASE REST APIs to manage whitelist, blacklist, API JSON files, and other settings.

### Dashboard

#### New in Dashboard 4.0.1

- **Load historical data to a Dashboard:** Add PingIntelligence data for a user specified time-frame to a new (or existing) Dashboard by updating the `dashboard.properties` file to enable fast-forward mode. For more information, see [Dashboard fast forward](#) on page 370

### Automated Deployment

#### New in PingIntelligence 4.0.1 automated deployment:

- **Install PingIntelligence components as a service:** Support automatic starting of PingIntelligence components upon a system restart. Configure `install_as_service` in the hosts file. For more information, see [Configure hosts file](#) on page 398
- **Use an existing MongoDB:** Use an existing customer deployed MongoDB instead of installing a new MongoDB for PingIntelligence. Configure `install_mongo` in hosts file. For more information, see [Configure hosts file](#) on page 398

- **ABS AI Engine to MongoDB SSL communication:** Enable SSL for ABS to MongoDB communication. Configure the `mongo_ssl` setting in `abs-defaults.yml` file. For more information, see [Change ABS default settings](#) on page 404.
- **Dashboard user setting:** Require restricted user access (that is, user cannot view sensitive content, such as token values) by configuring `restricted_user_access` in `dashboard-defaults.yml`. For more information, see [Change Dashboard default settings](#) on page 406

#### API Gateway Integration

**CA API Gateway integration:** PingIntelligence policy for CA API gateway is now available. For more information on gateway integration, see [PingIntelligence - CA API gateway sideband integration](#) on page 536

## PingIntelligence 4.0 Release Notes

---

- [ASE](#)
- [ABS](#)
- [Dashboard](#)

### ASE

#### New in ASE 4.0

- ASE running in sideband mode supports the capture of username information from the API management platform. For example, the PingAccess sideband policy passes PingFederate user information gathered from token introspection.
- Support for blocking of ABS AI Engine detected attacks based on usernames or API keys.
- Commands to add, remove usernames from ASE black list
- Improve sideband performance when configuring ASE with TCP keep-alive option for PingAccess and API gateways supporting a persistent TLS connection for sideband traffic.
- Single automated PingIntelligence deployment package for RHEL and Ubuntu environments.
- New CA, WSO2, and NGINX sideband policies released.

### ABS

#### New in ABS 4.0:

- User activity reports with information on APIs accessed, tokens/IPs used, URLs accessed, and other details.
- Forensic reporting on a specific user or API key activity including actions leading up to an attack.
- Customer controlled settings for retention (TTL) of black list entries by client identifier type – for example, block IP addresses for 7-days, block tokens for 1-day. Use the ABS Admin REST API to configure the retention period.
- Attack management tool for unblocking clients and adjusting AI engine thresholds
- AAD tool option to mirror discovered API definitions and updates on PingIntelligence
- Splunk alert notifications on Slack and email
- Support for log4j syslog output with option to tailor content for multiple SIEMs.
- Support for OpenJDK 11

### Dashboard

Updated PingIntelligence Dashboard with improved navigation and black list entries.

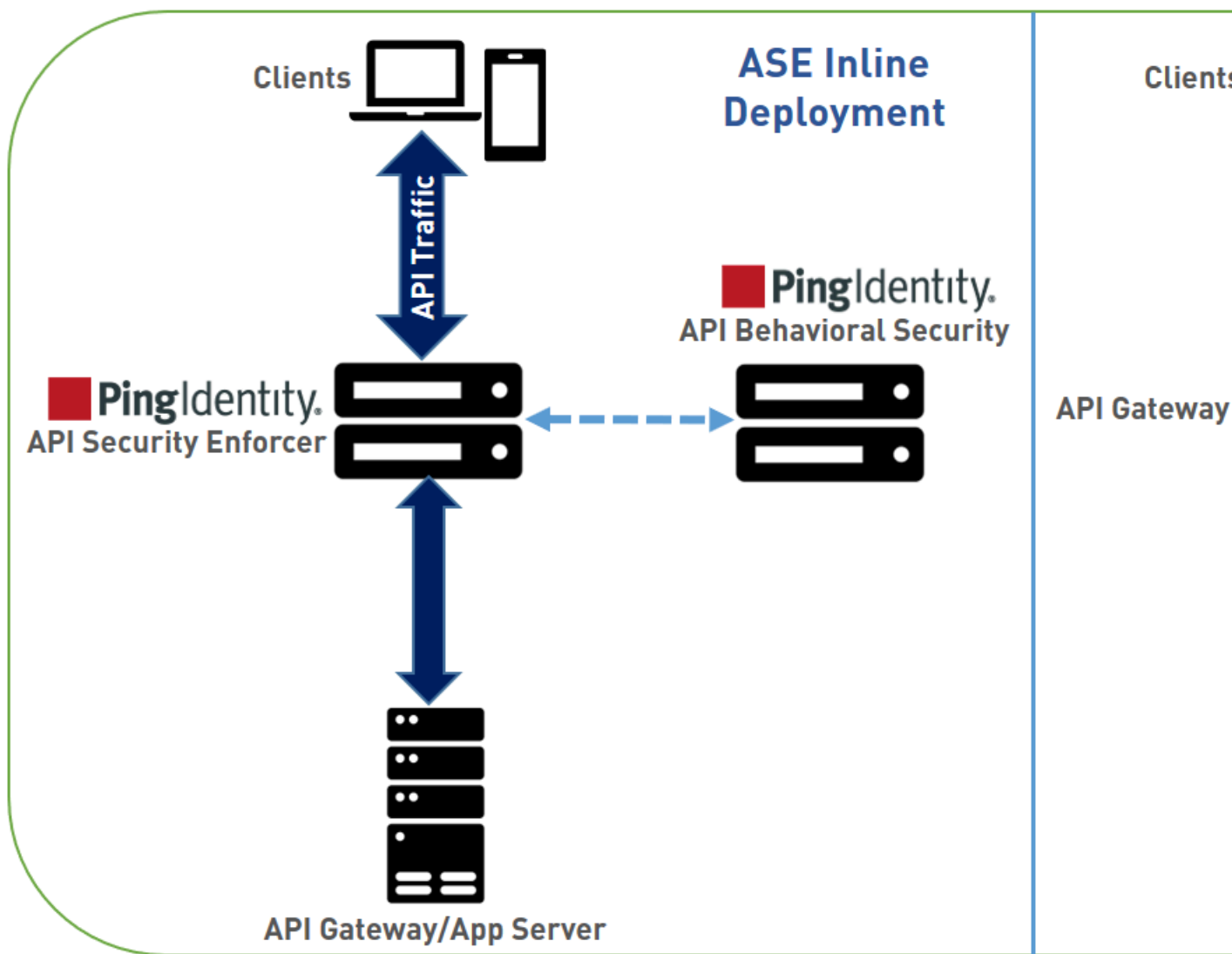
# API Security Enforcer

## Introduction

ASE supports multiple deployment 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.

<b>Security Features</b>	<b>Inline</b>	<b>Sideband</b>
<b>Interface to ABS AI Engine</b> for AI powered attack detection	Yes	Yes
<b>API deception</b> – decoy APIs look like legitimate APIs to hackers. After accessing a decoy API, a hacker is quarantined, plus activity information is collected.	Yes	Yes
<b>Real-time</b> 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
<b>Black and whitelist management</b> of tokens, API keys, cookies, IP addresses	Yes	Yes
<b>Real-time</b> blocking of API clients with traffic that deviates from API attributes.	Yes	No
<b>Dynamic mapping</b> of public API identity to private internal API identity	Yes	No
<b>Custom API error</b> messages prevent disclosure of sensitive error information.	Yes	No
<b>Admin Features</b>		
<b>Simple deployment</b> with modular JSON configuration files	Yes	Yes
<b>Live updates</b> – Add/remove without loss of traffic or stopping services.	Yes	Yes
<b>Obfuscation</b> – Keys and passwords are obfuscated	Yes	Yes
<b>Active-active clustering</b> – Supports scaling and resiliency: all nodes are peers and self-learn the configuration, traffic information, and security updates.	Yes	Yes
<b>Syslog information</b> messages sent to Syslog servers in RFC 5424 format.	Yes	Yes
<b>Automatic API discovery</b> discovers API JSON configuration data	Yes	Yes
<b>CLI and REST API</b> for management and automation tool integration.	Yes	Yes
<b>Linux PAM</b> -based administrator authentication with existing Linux tools.	Yes	Yes

<b>Audit log</b> captures administrative actions for compliance reporting.	Yes	Yes
<b>Distributed inbound flow</b> control limits client traffic and server traffic	Yes	No
<b>Multiprotocol Layer 7 routing</b> and load balancing of WebSocket, REST API	Yes	No
<b>Secure connection</b> 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.

#### 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.

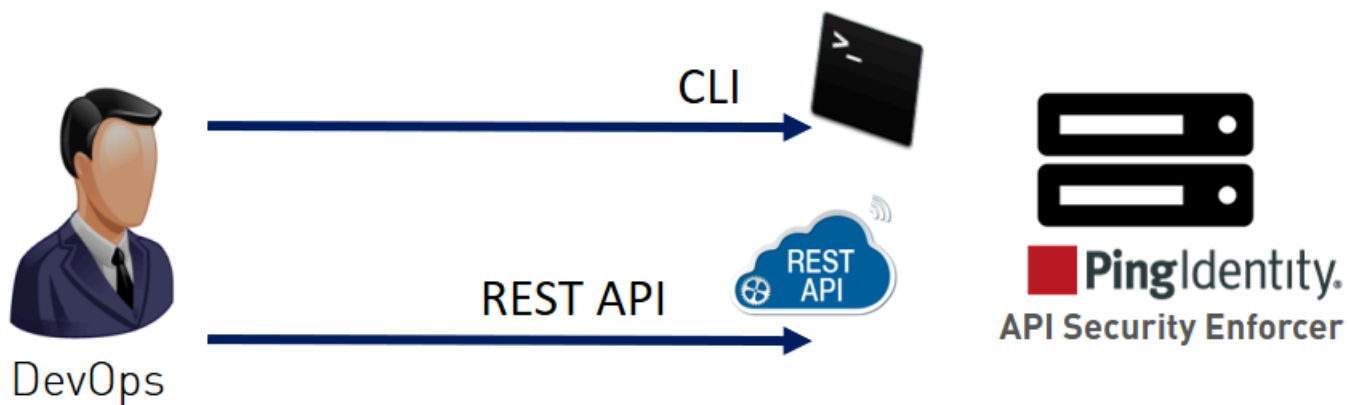
#### Update an existing license

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.

### 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 119 and [CLI for sideband ASE](#) on page 68

Option	Description
help	Displays <code>cli.sh</code> 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.

## 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]:
```

**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 admin
Old password >
New password >
New password again >
Password updated successfully
```

## Obfuscate keys and passwords

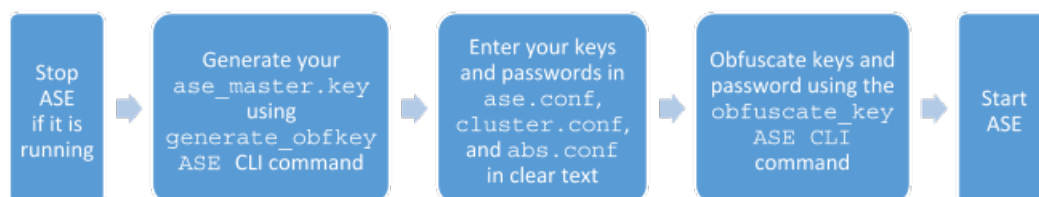
Using the ASE command line interface, you can obfuscate keys and passwords configured in `ase.conf`, `cluster.conf`, and `abs.conf`. Here is the obfuscated data in each file:

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

**i 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.

**i 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 <code>certs</code> 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 <code>abs_uploaded</code> 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.
	<p> <b>Note:</b> The scripts in the <code>bin</code> directory are not editable.</p>
util	The <code>util</code> directory contains scripts to check and open ABS ports as well as script to purge logs. <ul style="list-style-type: none"> <li>▪ <code>check_ports.sh</code> Check ABS ports</li> <li>▪ <code>open_ports_ase.sh</code>: Run this script on the ASE machine to open the default ASE ports: 80, 443, 8010, and 8020.</li> <li>▪ Purge logs</li> </ul>

## 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.

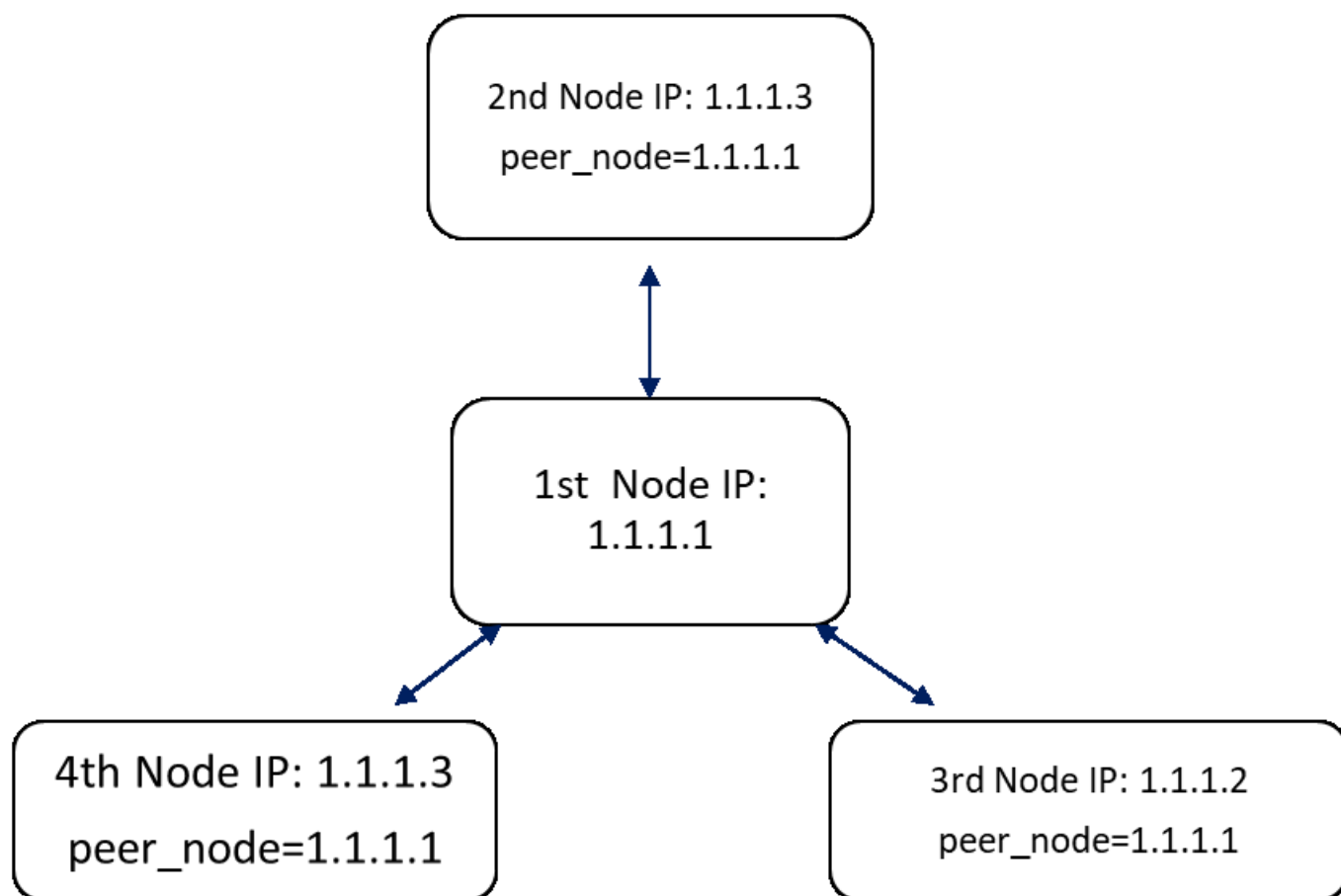
## ASE cluster deployment

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 <i>not</i> synchronized	<p>The following commands are <i>not</i> synchronized:</p> <ul style="list-style-type: none"> <li>▪ <code>create_key_pair</code></li> <li>▪ <code>create_csr</code></li> <li>▪ <code>create_self_sign_cert</code></li> <li>▪ <code>import_key_pair</code></li> <li>▪ <code>import_cert</code></li> <li>▪ <code>create_management_key_pair</code></li> <li>▪ <code>create_management_csr</code></li> <li>▪ <code>create_management_self_sign_cert</code></li> <li>▪ <code>import_management_key_pair</code></li> <li>▪ <code>import_management_cert</code></li> <li>▪ <code>update_password</code></li> <li>▪ <code>update_auth_method</code></li> <li>▪ <code>generate_obfkey</code></li> <li>▪ <code>obfuscate_keys</code></li> <li>▪ <code>update_keystore_password</code></li> </ul>	-
	<p><b>Note:</b> The commands listed above require the entire ASE to restart for the commands to synchronize.</p>	

### 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:7LFcqXQlqkW9kldQoFg0nJoLSojnzHDdbD3
```

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

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

### Post-install

Choose the first cluster node to run all CLI commands and REST API access for cluster consistency.

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

### 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 ASE cluster

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

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:

### 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
```

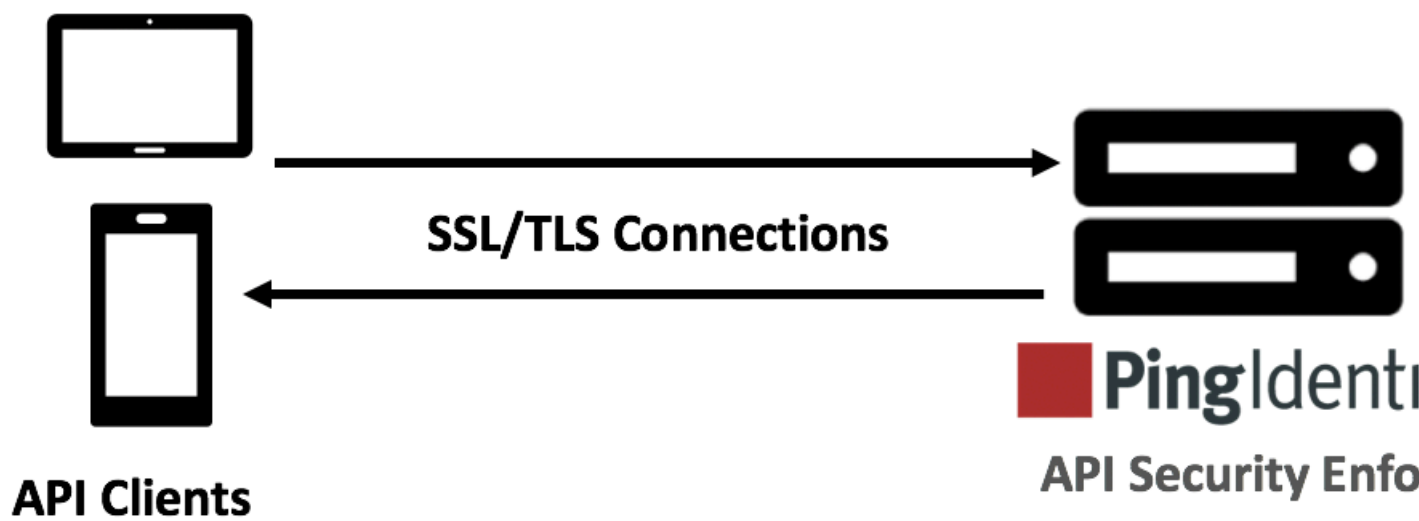
When the cluster stops, each cluster node retains all the cookie and certificate data.

## 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 `/opt/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
```

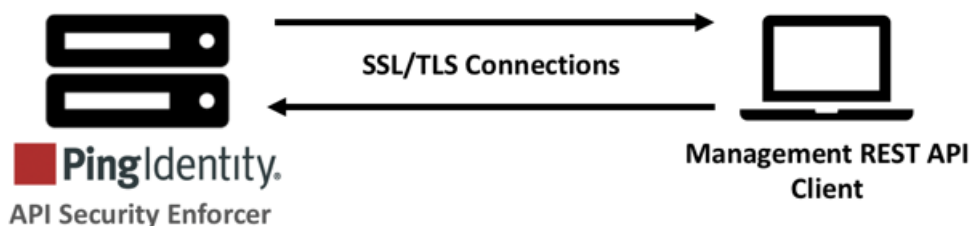
3. Restart ASE by stopping and starting.

## 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
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 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
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
```

## 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. 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-cert.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:

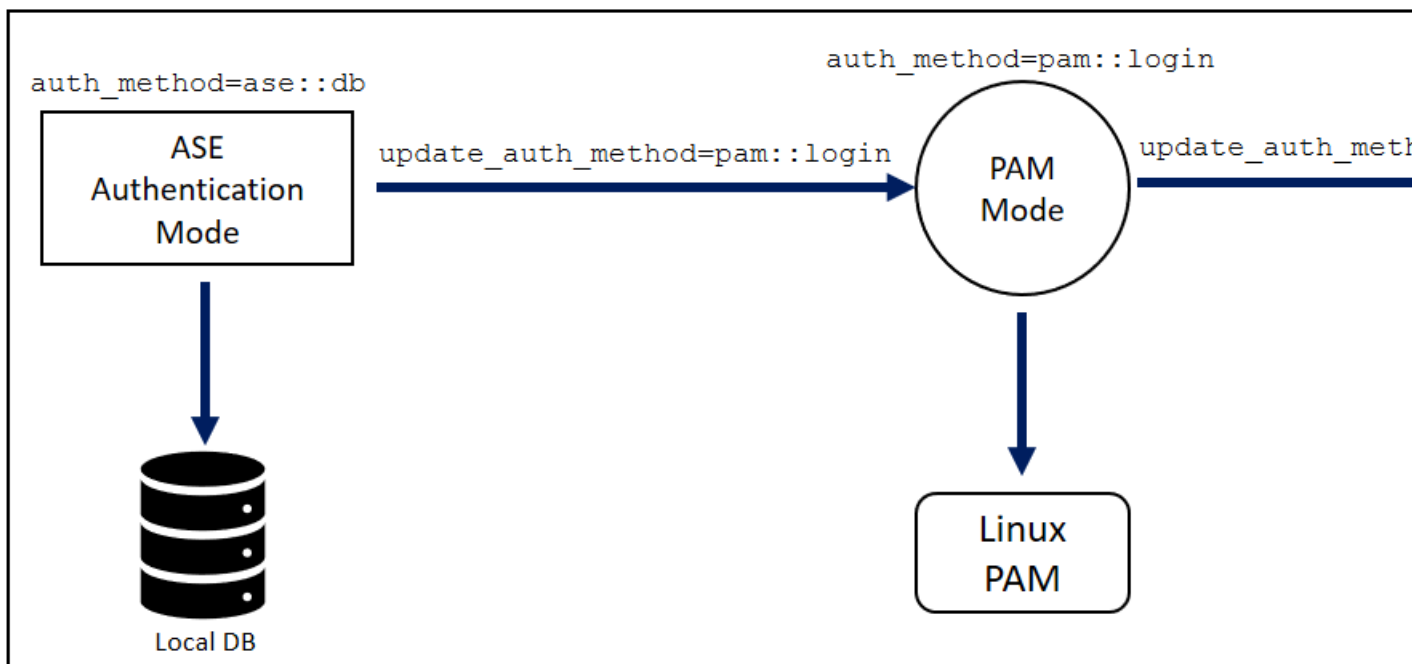
- Linux Pluggable Authentication Module (PAM)
- ASE native authentication (default method)

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

```

### 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 20.

```
; 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 occurred>
```

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@example.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@example.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 <code>ase.conf</code> file.

high CPU usage	<p><b>Cause:</b> 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 <code>ase.conf</code>, then ASE sends an alert.</p> <p><b>Resolution:</b> 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.</p>
high memory usage	<p><b>Cause:</b> 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 <code>ase.conf</code>, then ASE sends an alert.</p> <p><b>Resolution:</b> 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.</p>
high filesystem usage	<p><b>Cause:</b> 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 <code>ase.conf</code>, then ASE sends an alert.</p> <p><b>Resolution:</b> If ASE is reporting a high filesystem usage, check if the filesystem is getting full. Run the purge script available in the <code>util</code> directory to clear the log files.</p>
API added	<p>ASE sends an email alert when an API is added to ASE using CLI or REST API.</p> <p><b>Confirm:</b> 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.</p>
API removed	<p>ASE sends an email alert when an API is removed using CLI or REST API.</p> <p><b>Confirm:</b> ASE admin should verify whether the APIs were deleted intentionally or accidentally.</p>
API updated	<p>ASE sends an email alert when an API definition (the API JSON file) is updated by using CLI or REST API.</p> <p><b>Confirm:</b> ASE admin should verify whether the correct APIs was updated.</p>
Server added	<p>ASE sends an email alert when a server is added to an API by using CLI or REST API.</p> <p><b>Confirm:</b> ASE admin should verify whether the correct server was added to API.</p>
Server removed	<p>ASE sends an email alert when a server is removed from an API by using CLI or REST API.</p> <p><b>Confirm:</b> ASE admin should verify whether the correct server was removed from an API.</p>
cluster node up	<p>ASE sends an email alert when a node joins an ASE cluster.</p> <p><b>Confirm:</b> ASE admin should verify whether the correct ASE node joined the ASE cluster.</p>



cluster node down	ASE sends an email alert when a node is removed from an ASE cluster. <b>Confirm:</b> 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. <b>Resolution:</b> ASE admin should investigate the reason for the backend API server being not reachable from ASE. You can run the ASE <code>health_status</code> 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.

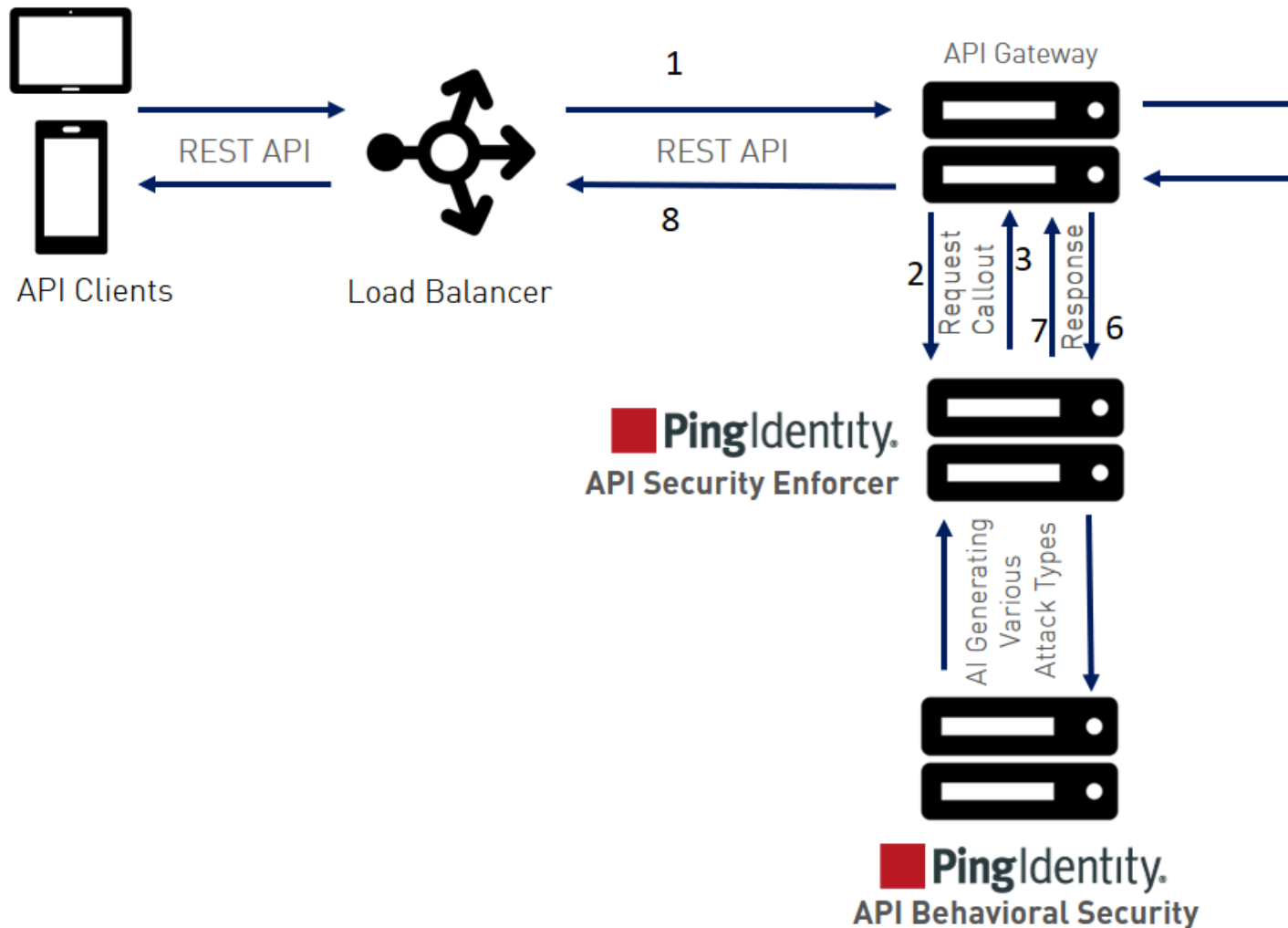
Email alert	Possible cause and resolution
Network error	<b>Cause:</b> ABS IP may not be reachable or ASE is not able to connect ABS IP and port. <b>Resolution:</b> <ul style="list-style-type: none"> <li>▪ If there is a firewall in the deployment, check whether firewall is blocking access to ABS.</li> <li>▪ Check whether ABS is running.</li> <li>▪ Check whether correct IP address is provided in the <code>abs.conf</code> file.</li> </ul>
ABS seed node resolve error	<b>Cause:</b> The hostname provided in <code>abs.conf</code> could not be resolved. <b>Resolution:</b> Check whether correct IP address is provided in <code>abs.conf</code> file.
ABS SSL handshake error	<b>Cause:</b> SSL handshake error could be because of an invalid CA certificate. <b>Resolution:</b> Check whether a valid CA certificate is configured in ASE.
ABS authentication error	<b>Cause:</b> Authentication error could be because of invalid access and secret key. <b>Resolution:</b> Confirm the access key and secret key configured is the same that is configured in ABS <code>abs.properties</code> file.
ABS cluster info error	<b>Cause:</b> Error while fetching ABS cluster information. <b>Resolution:</b> Check the <code>controller.log</code> file.
ABS config post error	<b>Cause:</b> Error while sending API JSON definition to ABS <b>Resolution:</b> Check the <code>controller.log</code> file.

ABS service unavailable error	<b>Cause:</b> ABS returning 503 response code. <b>Resolution:</b> Check the <code>abs.log</code> file.
Log upload error	<b>Cause:</b> API call to upload access log files to ABS fails. <b>Resolution:</b> Check both ASE's <code>controller.log</code> and ABS <code>abs.log</code> file.
Duplicate log upload error	This is an informative message.
ABS node queue full error	<b>Cause:</b> 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. <b>Resolution:</b> Increase the number of ABS nodes.
ABS node capacity low error	<b>Cause:</b> ABS resources are utilized to a maximum. <b>Resolution:</b> Increase the number of ABS nodes.
ABS attack get error	<b>Cause:</b> Error while fetching attack list from ABS <b>Resolution:</b> Check ASE's <code>controller.log</code> 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
<b>ASE mode</b>	
<code>mode</code>	Change the <code>mode</code> to <code>sideband</code> for ASE to work in a sideband mode. The default value is <code>inline</code> .
<code>enable_sideband_keepalive</code>	When set to <code>true</code> , ASE sends a keep-alive in response header for the TCP connection between API gateway and ASE. With the default <code>false</code> value, ASE sends a connection close in response header for connection between API gateway and ASE.
	<div style="border: 1px solid black; padding: 5px;"> <p><b>Note:</b> This parameter is applicable only when <code>mode</code> is set to <code>sideband</code>.</p> </div>
<code>enable_sideband_authentication</code>	This parameter only applies in the ASE sideband mode. Set it to <code>true</code> to enable authentication between in client, for example, an API gateway and ASE. After setting it to <code>true</code> , generate a sideband authentication token using ASE <code>create_sideband_token</code> command.
<b>ASE ports</b>	

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.
<b>ASE administration and audit</b>	
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 <a href="#">Configuring Native and PAM Authentication</a> for more information on the two options <ul style="list-style-type: none"> <li>▪ ase::db (Default - Native authentication)</li> <li>▪ pam::ldap (Linux-PAM authentication with script)</li> </ul>
ase_health	When true, enables load balancers to perform a health check using the following URL: "http(s)://<ASE Name>/ase" where <ASE Name> is the ASE domain name The default value is false.
<p> <b>Note:</b> Do not configure the /ase URL in an API JSON file.</p>	
enable_1G	N/A
http_ws_process	The number of HTTP processes. It is set to 1. Do not change this value.
https_wss_process	The number of HTTPS or processes. It is set to 1. Do not change this value.
enable_access_log	When true, log client traffic request and response information. Default value is true.
flush_log_immediate	When true, log files are immediately written to the file system. When false, log files are written after a time interval. The default value is true.
attack_list_memory	The amount of memory used for maintaining black and whitelists. The default value is 128 MB.

**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 and detecting attacks using machine learning algorithms.

enable\_abs\_attack

When `true`, ASE fetches attack list from ABS and blocks access by clients in the attack list.

When `false`, attack list is not downloaded.

abs\_attack\_request\_minute

Time interval in minutes at which ASE fetches ABS attack list. The default value is 10 minutes.

### Alerts and reports

enable\_email

When `true`, send email notifications. See [Email alerts and reports](#) on page 35 for more information. The default value is `false`.

email\_report

Time interval in days at which ASE sends reports. Minimum value is one day and the maximum is seven days.

The default value is 1.

smtp\_host

Hostname of SMTP server.

smtp\_port

Port number of SMTP server.

smtp\_ssl

Set to `true` if you want email communication to be over SSL. Make sure that the SMTP server supports SSL. If you set `smtp_ssl` to `true` and the SMTP server does not support SSL, email communication falls back to 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	<p>Set to <code>true</code> if you want ASE to verify the SMTP server's SSL certificate. The default value is <code>true</code>.</p> <p>If you set it to <code>false</code>, ASE does not verify SMTP server's SSL certificate; however, the communication is still over SSL.</p> <div data-bbox="779 325 1622 451" style="border: 1px solid black; padding: 5px;"> <p><b>Note:</b> If you have configured an IP address as <code>smtp_host</code> and set <code>smtp_cert_verification</code> to <code>true</code>, then make sure that the certificate configured on the SMTP server has the following:</p> <pre style="margin: 0;">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</pre> </div>
sender_email	Email address for sending email alerts and reports.
sender_password	Password of sender's email account.
receiver_email	<p>Email address to notify about alerts and reports</p> <p>See <a href="#">email alerts</a> for more information.</p>
<b>ASE server resource utilization</b>	
cpu_usage	<p>Percentage threshold value of CPU utilization.</p> <p>See <a href="#">email alerts</a> for more information.</p>
memory_usage	<p>Percentage threshold value of memory usage.</p> <p><a href="#">email alerts</a> alerts for more information.</p>
filesystem_size	<p>Percentage threshold value of filesystem capacity.</p> <p>See <a href="#">email alerts</a> for more information.</p>
buffer_size	<p>Customizable payload buffer size to reduce the number of iterations required for reading and writing payloads.</p> <p>Default value is 16KB. Minimum is 1KB and maximum is 32KB.</p>

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:sRNp0W7sSilzrReXeHodKQ:1XcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerM

```

### 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”.
- There should not be any spaces in the name of the API.

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	<p>The value of the URL for the managed API. You can configure up to three levels of sub-paths. For example,</p> <p>"/shopping"- name of a 1 level API</p> <p>"/shopping/electronics/phones" -3 level API</p> <p>"/" – entire server (used for ABS API Discovery or load balancing)</p>
hostname	<p>Hostname for the API. The value cannot be empty.</p> <p>"*" matches any hostname.</p>
<p><b>Configure the client identifiers (for example, cookie, API key, OAuth2 token) used by the API</b></p>	
cookie	Name of cookie used by the backend servers.
cookie_idle_timeout	N/A
logout_api_enabled	
cookie_persistence_enabled	
oauth2_access_token	<p>When <code>true</code>, ASE captures OAuth2 Access Tokens.</p> <p>When <code>false</code>, ASE does not look for OAuth2 Tokens.</p> <p>Default value is <code>false</code>.</p> <p>For more information, see <a href="#">Configuring OAuth2 Token</a>.</p>
apikey_qs	<p>When API key is sent in the query string, ASE uses the specified parameter name to capture the API key value.</p> <p>For more information, see <a href="#">Configuring API keys</a>.</p>
apikey_header	<p>When API key is part of the header field, ASE uses the specified parameter name to capture the API key value.</p> <p>For more information, see <a href="#">Configuring API keys</a>.</p>
login_url	Public URL used by a client to connect to the application.
enable_blocking	<p>When <code>true</code>, ASE blocks all types of attack on this API. When <code>false</code>, no attacks are blocked.</p> <p>Default value is <code>false</code>.</p>
api_mapping	N/A

<b>API pattern enforcement</b>	N/A
protocol_allowed	
http_redirect	
methods_allowed	
content_type_allowed	
error_code	
error_type	
error_message_body	
<b>Flow control</b>	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.
<b>Health check</b>	N/A
health_check_interval	
health_retry_count	
health_url	
server_ssl	N/A
<b>Servers:</b>	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	
<b>Decoy Config</b>	When <code>decoy_enabled</code> is set to <code>true</code> , decoy sub-paths function as decoy APIs .
decoy_enabled	
response_code	<code>response_code</code> is the status code (for example 200) that ASE returns when a decoy API path is accessed.
response_def response_message	<code>response_def</code> is the response definition (for example OK) that ASE returns when a decoy API path is accessed.
decoy_subpaths	<code>response_message</code> is the response message (for example OK) that ASE returns when a decoy API path is accessed. <code>decoy_subpaths</code> is the list of decoy API sub-paths (for example <code>shop/admin, shop/root</code> ) See <a href="#">Configuring API deception</a> for details

Here is a sample JSON file for a REST API:

```
{ "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": "",
  "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",
  "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": [
```

```
]
}
}
}
```

**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 ASE API JSON. PingIntelligence AI engine identifies the username 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

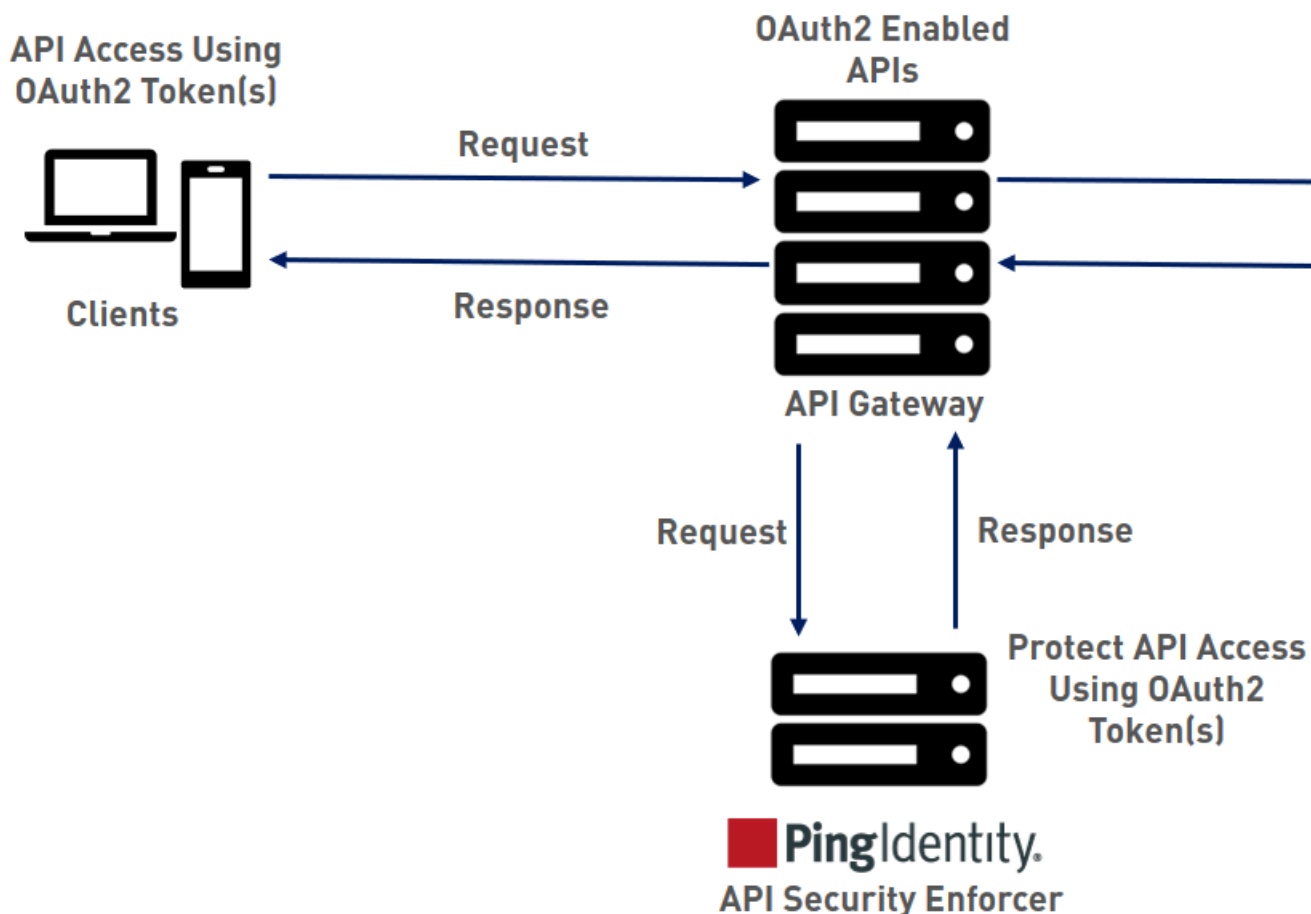
ASE supports capturing and blocking of OAuth2 tokens. To enable OAuth2 token capture, set the value of `oauth2_access_token` to `true` in the API JSON file. Here is a snippet of an API JSON file with OAuth2 token capture activated. To disable, change the value to `false`.

```
"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.

## 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 : pZ1hg5s3i8csImMoas7vh81vz
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.

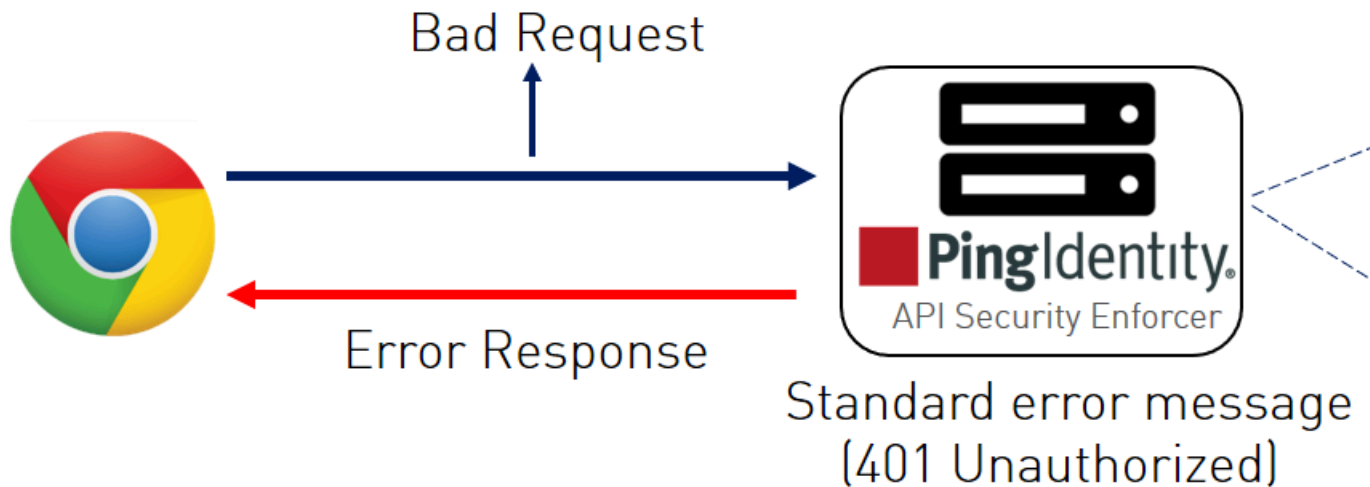
**ASE generated error messages for blocked requests**

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 Hostname
Malformed Request	400	Bad Request	Error: Malformed Request
IP attack	403	Unauthorized	Error: Unauthorized
Cookie attack	403	Unauthorized	Error: Unauthorized
OAuth2 Token attack	403	Unauthorized	Error: Unauthorized
API Key attack	403	Unauthorized	Error: Unauthorized
Username attack	403	Unauthorized	Error: Unauthorized

The con  
the



### 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

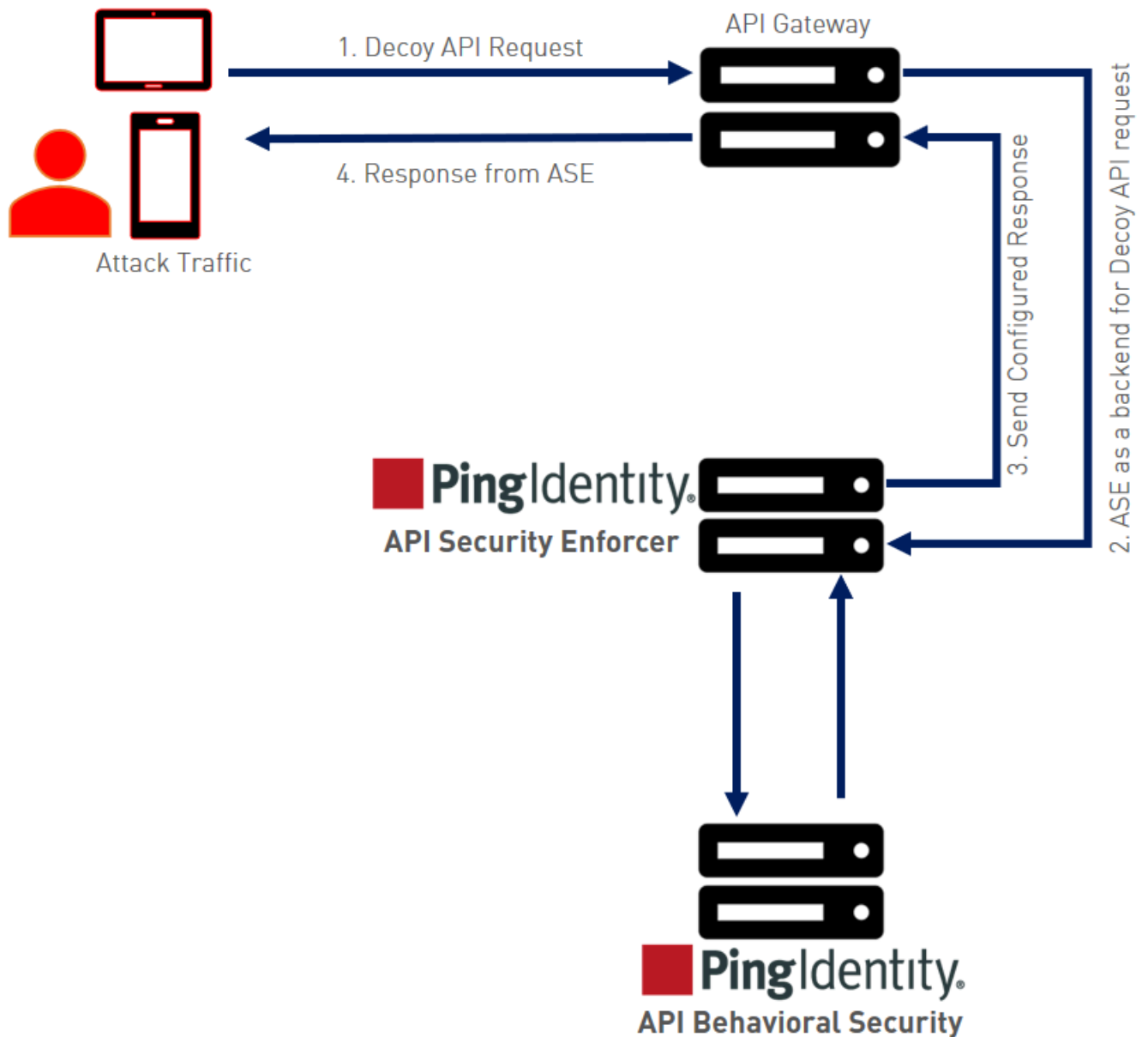
A decoy API is configured in ASE and the API gateway. It requires no changes to backend servers. It appears as part of the API ecosystem and is used to detect the attack patterns of hackers. When a hacker accesses a decoy API, ASE sends a predefined response (defined in the `response_message` parameter 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 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:

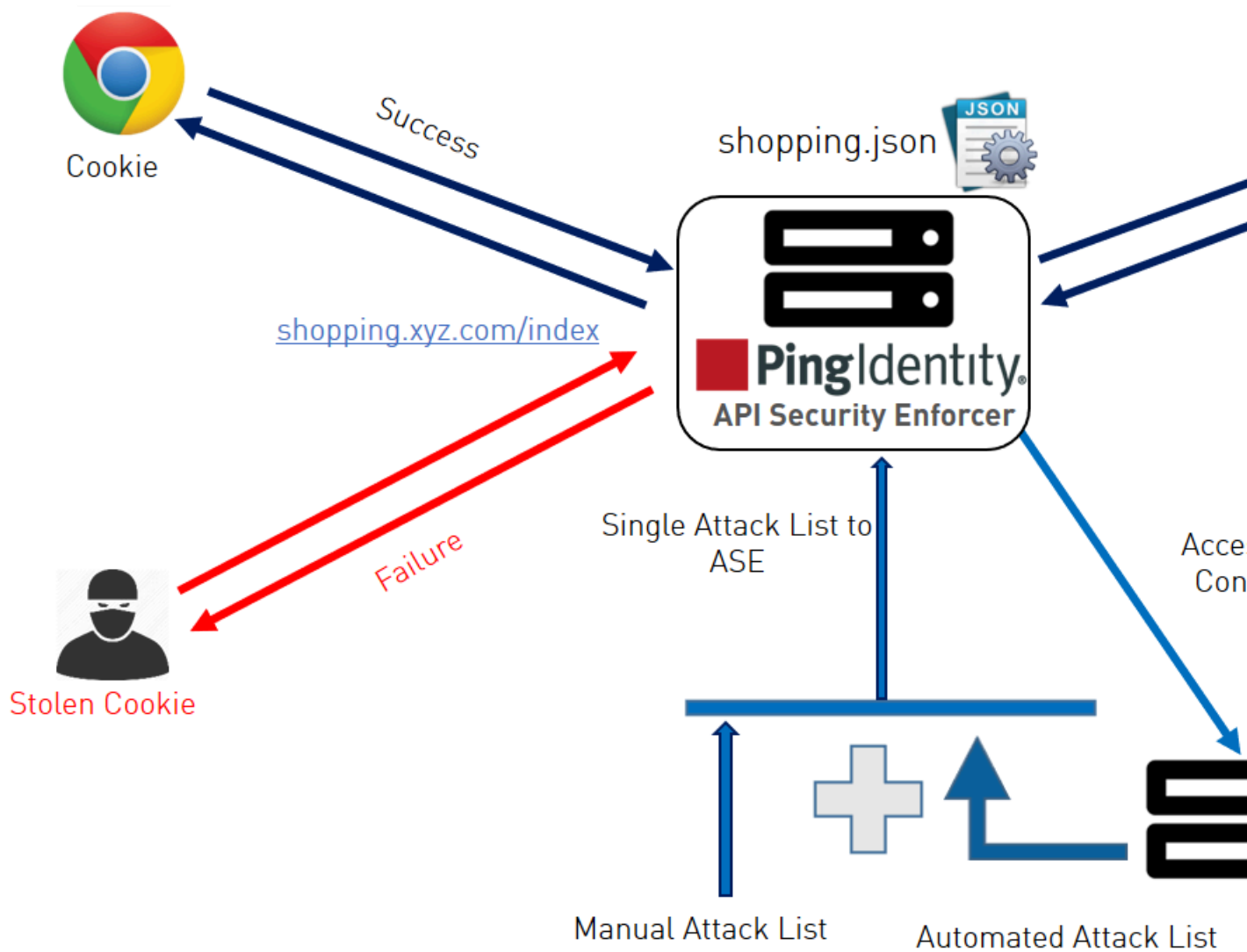
```
[Tue Aug 1422:51:49:707 2018] [thread:209] [info] [connectionid:1804289383]
[connectinfo:100.100.1.1:36663] [type:connection_drop] [api:decoy]
[request_payload_length:0] GET /decoy/test/test HTTP/1.1
User-Agent: curl/7.35.0
Accept: */*
Host: app
```

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
<code>abs_endpoint</code>	Hostname and port or the IPv4 and port of all the ABS nodes
<code>access_key</code>	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

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 MongoDB database. This value is obfuscated during the start of ASE.
enable_ssl	Set the value to true for SSL communication between ASE and ABS. The default value is true. ASE sends the access log files in plain text if the value is set to false.
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 still 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, enter 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 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
```

### 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/pingidentity/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_attacks
```

**Disable ASE Detected Attacks****Description**

Disable API firewall

**Syntax**

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

**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

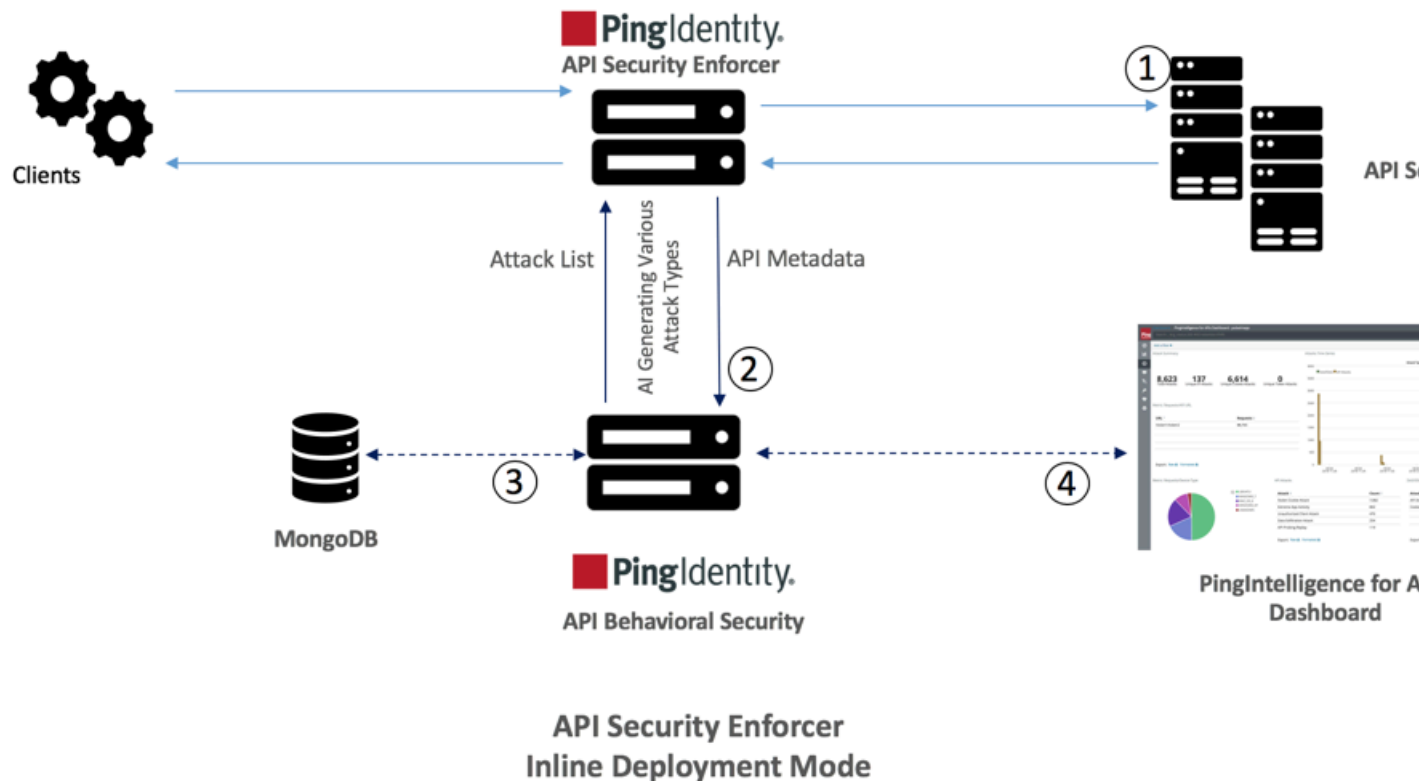
#### Syntax

```
./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.



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
<b>ASE mode</b>	
<code>mode</code>	The mode in which ASE works. Possible values are <code>inline</code> and <code>sideband</code> . The default value is <code>inline</code> .
<code>enable_sideband_keepalive</code>	NA
<code>enable_sideband_authentication</code>	NA
<b>ASE ports</b>	
<code>http_ws_port</code>	Data port used for http or WebSocket protocol. The default value is 80.
<code>https_wss_port</code>	Data port used for https or Secure WebSocket (wss). The default value is 443.

management_port	<p>Management port used for CLI and REST API management.</p> <p>The default value is 8010.</p>
	<p><b>ASE administration and audit</b></p>
admin_log_level	<p>The level of log detail captured. Options include:</p> <p>Fatal – 1, Error – 2, Warning – 3, Info – 4, Debug – 5</p>
enable_audit	<p>When set to true, ASE logs all actions performed in ASE in the audit log files.</p> <p>The default value is true.</p>
syslog_server	<p>Syslog server hostname or IPv4 address:port number.</p> <p>Leave this parameter blank if you do not want to generate for no syslog.</p>
hostname_refresh	<p>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.</p>
auth_method	<p>Authentication method used for administrator access. See <a href="#">Configuring Native and PAM Authentication</a> for more information on the two options.</p> <ul style="list-style-type: none"> <li>ase::db (Default - Native authentication)</li> <li>pam::ldap (Linux-PAM Authentication with script)</li> </ul>
enable_ase_health	<p>When <code>true</code>, enables load balancers to perform a health check using following URL: <code>http(s)://&lt;ASE Name&gt;/ase</code> where <code>&lt;ASE Name&gt;</code> is the domain name</p> <p>The default value is <code>false</code>.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> Do not configure the /ase URL in an API JSON file.</p> </div>
enable_1G	<p>When <code>true</code>, enable 1Gbps Ethernet support.</p> <p>The default value is <code>true</code>.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> Only applicable when using a 1G NIC card</p> </div>
http_ws_process	<p>The number of HTTP or WebSocket processes.</p> <p>The default value is 1 and the maximum value is 6.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> When running ASE in a cluster deployment, all nodes must have the same number of processes.</p> </div>
https_wss_process	<p>The number of HTTPS or secure WebSocket processes.</p> <p>The default value is 1 and the maximum value is 6.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> When running ASE in a cluster deployment, all nodes must have the same number of processes.</p> </div>

enable_access_log	When <code>true</code> , log client traffic request and response information. Default value is <code>true</code> .
flush_log_immediate	When <code>true</code> , log files are immediately written to the file system. When <code>false</code> , log files are written after a time interval. The default value is <code>true</code> .
attack_list_memory	The amount of memory used for maintaining black and whitelists. The default value is 128 MB.
keystore_password	Password for the keystore. For more information on updating the keystore password, see <a href="#">Updating Keystore Password</a> .
<b>ASE cluster</b>	
enable_cluster	When <code>true</code> , run the setup in cluster mode. The default value is <code>false</code> , run the setup in standalone mode.
<b>Security</b>	
enable_sslv3	When <code>true</code> , enable SSLv3. Default value is <code>false</code> .
server_ca_cert_path	Location of the trusted CA certificates for SSL/TLS connections from ASE to backend servers.  If the path parameter value is left empty, then ASE does not verify the validity of CA certificates. However, the backend connection is still encrypted.  For RHEL 7.6 CA certificates, the default path is: <code>/etc/pki/tls/certs/</code> .  Multiple certificates can be placed in this directory.
enable_xff	When <code>true</code> , pass XFF header with originating IP address to the backend server.
enable_firewall	When <code>true</code> , activate the following API security features: <ul style="list-style-type: none"> <li>▪ API mapping</li> <li>▪ API pattern enforcement</li> <li>▪ Connection drop using attack types</li> <li>▪ Flow control</li> </ul> Default value is <code>true</code>
enable_ase_detected_attack	<b>Real-time API security</b> When <code>true</code> , activates the real-time security in ASE. ASE detects and blocks pattern enforcement violations, wrong API keys and clients proxying to decoy API and later accessing real APIs. The default value is <code>false</code> .
<b>API deception</b>	
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).
<b>AI-based API security (ABS)</b>	
enable_abs	When <code>true</code> , send access log files to ABS for generating API metrics and detecting attacks using machine learning algorithms.

enable_abs_attack	<p>When <code>true</code>, ASE fetches attack list from ABS and blocks access by the clients that are in the attack list.</p> <p>When <code>false</code>, attack list is not downloaded.</p>
abs_attack_request_minute	<p>Time interval in minutes at which ASE fetches ABS attack list. The default value is 10-minutes.</p>
<b>Alerts and reports</b>	
enable_email	<p>When <code>true</code>, send email notifications. The default value is <code>false</code>. ASE logs the alerts in <code>balancer.log</code> file even when email alerts are disabled. See <a href="#">Email alerts and reports</a> on page 35 for more information.</p>
email_report	<p>Time interval in days at which ASE sends reports. Minimum value is 1-day and the maximum is 7-days. The default value is 1-day.</p>
smtp_host	<p>Hostname of SMTP server.</p>
smtp_port	<p>Port number of SMTP server.</p>
smtp_ssl	<p>Set to <code>true</code> if you want email communication to be over SSL. Make sure that the SMTP server supports SSL. If you set <code>smtp_ssl</code> to <code>true</code> and SMTP server does not support SSL, email communication falls back to a non-SSL channel. The default value is <code>true</code>.</p> <p>Set it to <code>false</code> if email communication is over a non-SSL channel. The communication will fail if you set the parameter to <code>false</code>, but the SMTP server only supports SSL communication.</p>
smtp_cert_verification	<p>Set to <code>true</code> if you want ASE to verify the SMTP server's SSL certificate. The default value is <code>true</code>.</p> <p>If you set it to <code>false</code>, ASE does not verify SMTP server's SSL certificate; however, the communication is still over SSL.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p><b>Note:</b> If you have configured an IP address as <code>smtp_host</code> and set <code>smtp_cert_verification</code> to <code>true</code>, then make sure that the certificate configured on the SMTP server has the following:</p> <pre style="margin: 0;">X509v3 extensions:     X509v3 Key Usage:         Key Encipherment, Data Encipherment     X509v3 Extended Key Usage:         TLS Web Server Authentication     X509v3 Subject Alternative Name:         <b>IP Address: X.X.X.X</b></pre> </div>
sender_email	<p>Email address for sending email alerts and reports.</p>
sender_password	<p>Password of sender's email account.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p><b>Note:</b> You can leave this field blank if your SMTP server does not require authentication.</p> </div>
receiver_email	<p>Email address to notify about alerts and reports</p> <p>See <a href="#">email alerts</a> for more information.</p>

**ASE server resource utilization**

cpu_usage	Percentage threshold value of CPU utilization. See <a href="#">email alerts</a> for more information.
memory_usage	Percentage threshold value of memory usage. See <a href="#">email alerts</a> for more information.
filesystem_size	Percentage threshold value of filesystem capacity. See <a href="#">email alerts</a> 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:sRNp0W7sSilzrReXeHodKQ:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerM

```

### 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”.
- There should not be any spaces in the name of the API.

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 ; http
url	The value of the URL "/shopping"- name "/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 form s: seconds, m: minute <ul style="list-style-type: none"> <li>▪ w: week</li> <li>▪ mnt: month</li> <li>▪ yr: year</li> </ul>
logout_api_enabled	When <code>true</code> , ASE exp
cookie_persistence_enabled	When <code>true</code> , the sub
oauth2_access_token	When <code>true</code> , ASE cap When <code>false</code> , ASE d For more information,
apikey_qs	When API Key is sent For more information,
apikey_header	When API Key is part For more information,
login_url	Public URL used by a
enable_blocking	When <code>true</code> , ASE blo Default value is false.

api\_memory\_size

Maximum ASE mem

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

```
"health_check_h
  "X-
  "X-
  },
```

### Example

#### Example Key

X-Host

X-Custom-Header

server\_ssl

When set to true, ASE  
server.

#### Servers:

host

The IP address or hos

See [REST API Prote](#)

port

server\_spike\_threshold

server\_connection\_quota

**API Mapping:**

internal\_url

The following API Pattern Enforcement parameters only apply when API Firewall is activated

**Flow Control**

client\_spike\_threshold

server\_connection\_queueing

bytes\_in\_threshold

bytes\_out\_threshold

protocol\_allowed

http\_redirect

response\_code

response\_def

https\_url

methods\_allowed

content\_type\_allowed

error\_code

error\_type

error\_message\_body

**Decoy Config**

decoy\_enabled

response\_code

response\_def response\_message

decoy\_subpaths

Internal URL is mapped

See [API Name Mapping](#)

ASE flow control ensures

See [WebSocket API](#)

List of accepted protocols

Values can be HTTP, HTTPS

**Note:** When Firewall is enabled, the following parameters are not applicable:

Redirect unencrypted traffic

See [Configuring Patterns](#)

List of accepted REST methods

GET, POST, PUT, PATCH, DELETE

List of content types allowed

Error message generated

See [ASE Detected Errors](#)

When decoy\_enabled is true,

response\_code is the response code

response\_def is the response definition

response\_message is the response message

decoy\_subpaths is the list of subpaths

See [Configuring API Decoy](#)

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": {
  "X-Host": "%{HOST}",
  "X-Custom-Header": "value"
},
"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": [
  ]
}
}
}

```

### 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
```

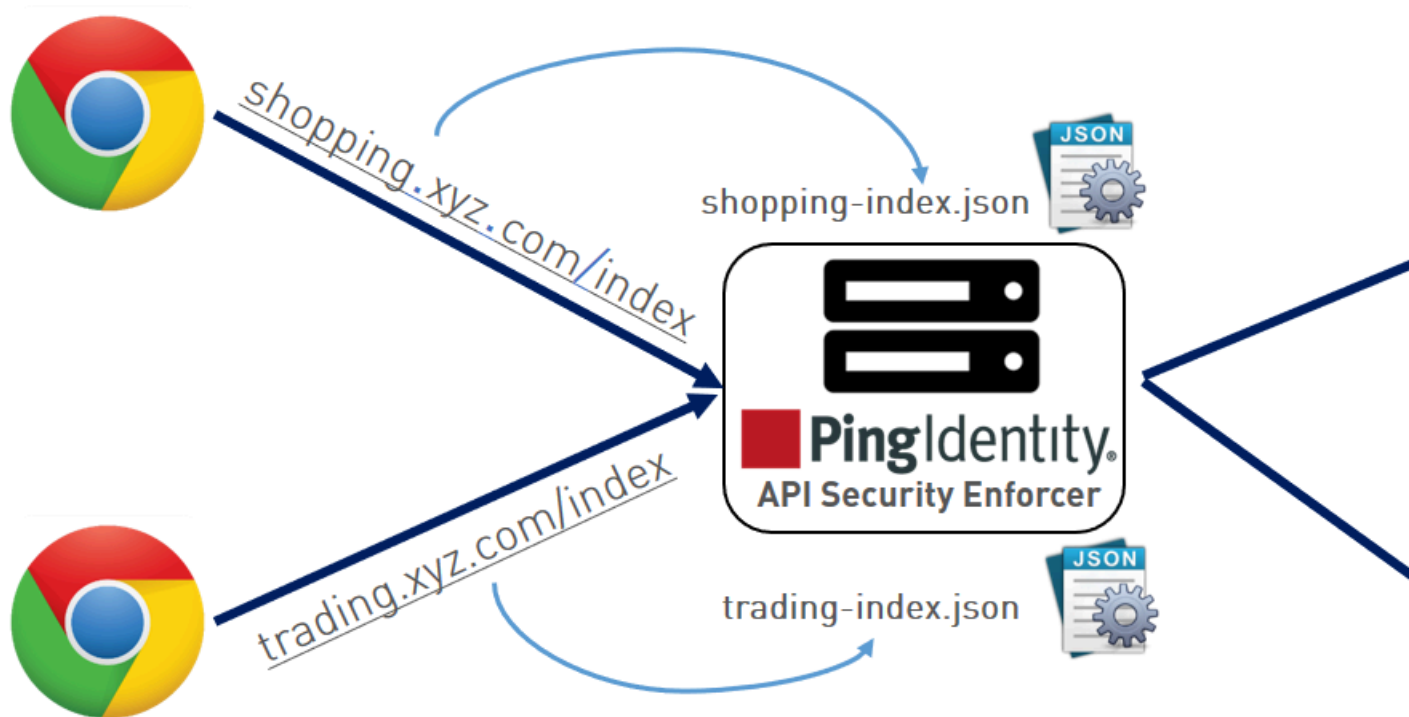
## API routing

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,
```

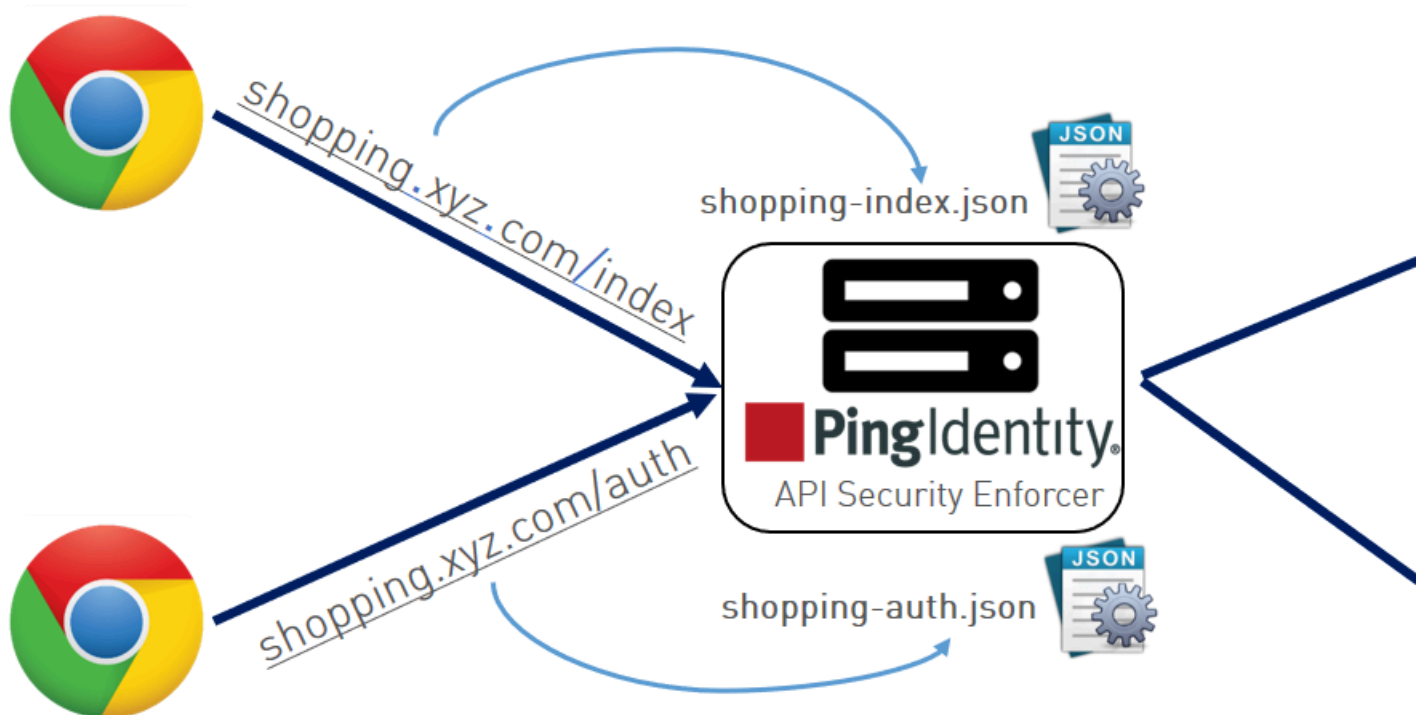
For each API, configure a separate JSON file.



### 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`.

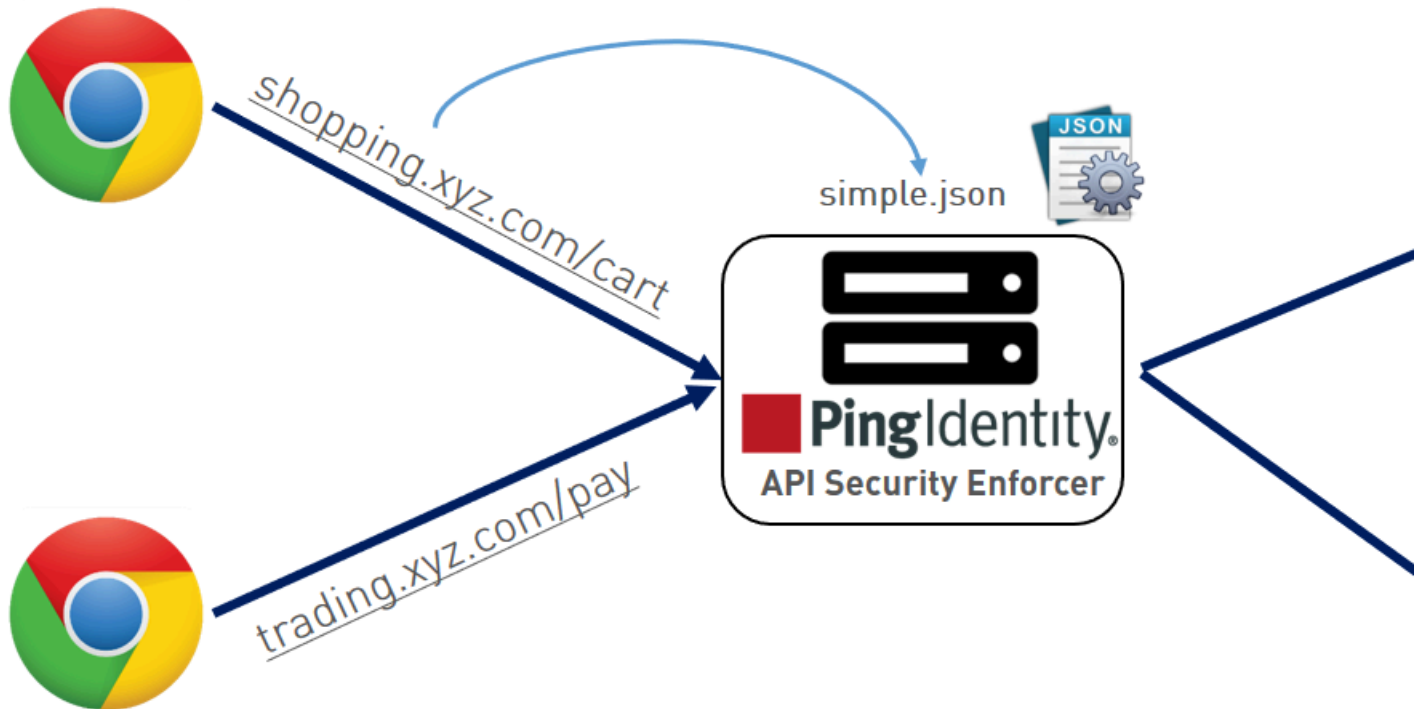


### Wildcard hostname and API name

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 103)

## 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 OauthTokenusemethodid12345
[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 OauthTokenusemethodid12345
```

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

The following sections describe how to configure ASE to capture OAuth2 Tokens and API keys.

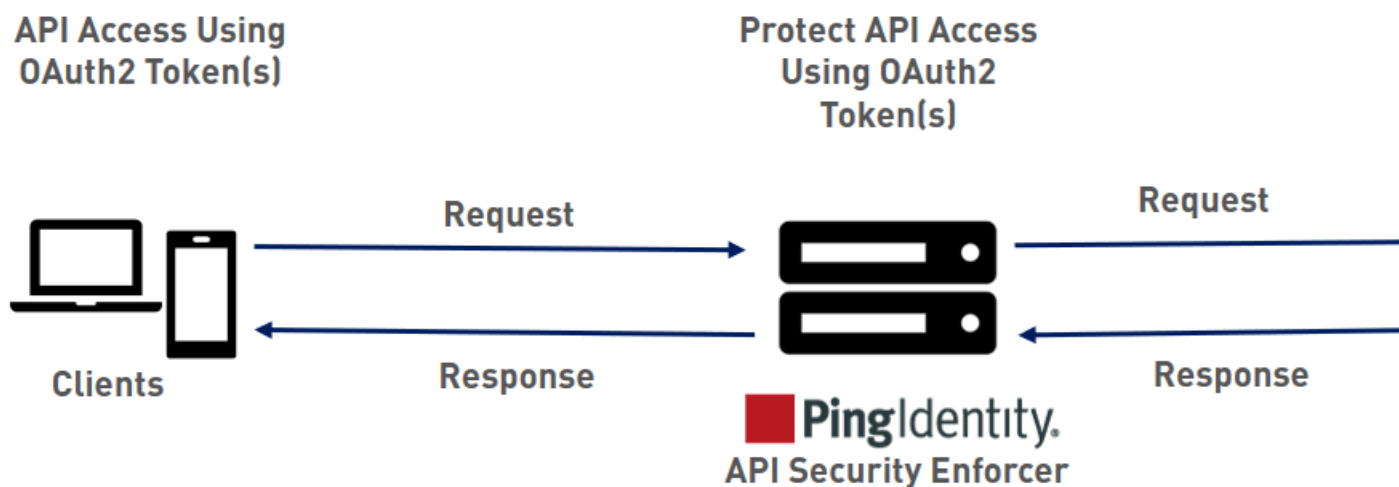
### Configure ASE support for OAuth2 tokens

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.

### 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
Realtme 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
Realtme 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
Realtme 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

```

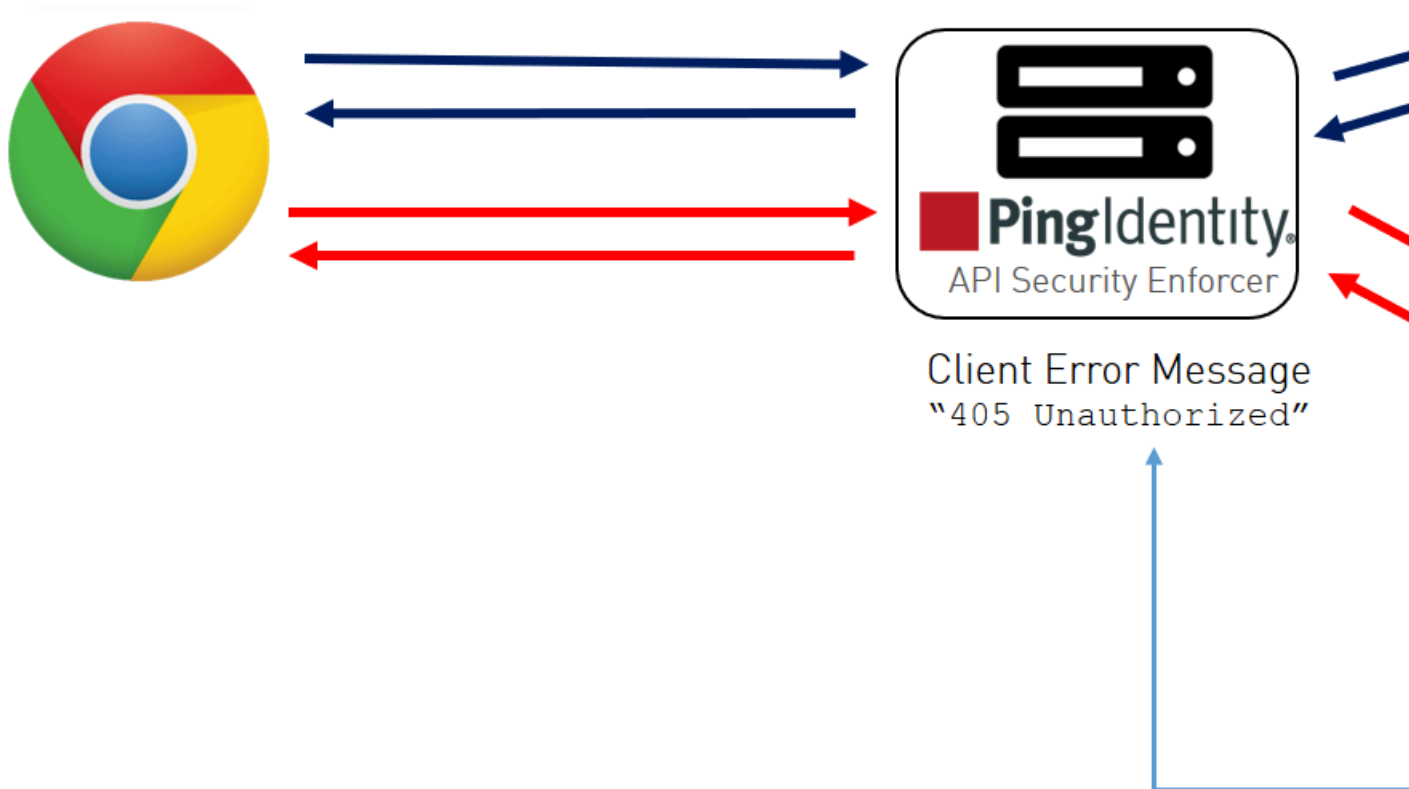
./bin/cli.sh -u admin -p admin clear_blacklist
This will delete all blacklist Attacks, Are you sure (y/n) :y
Blacklist cleared
./bin/cli.sh -u admin -p admin clear_blacklist
This will delete all blacklist Attacks, Are you sure (y/n) :n
Action canceled

```

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.

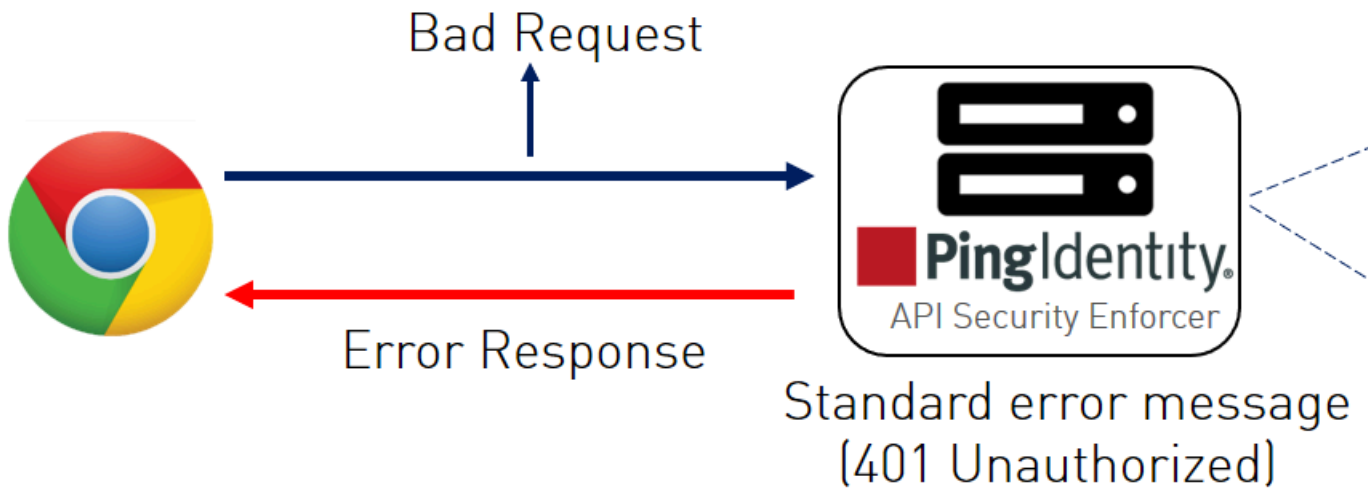
### ASE generated error messages for blocked requests

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 Hostname
Malformed Request	400	Bad Request	Error: Malformed Request
IP attack	403	Unauthorized	Error: Unauthorized
Cookie attack	403	Unauthorized	Error: Unauthorized
OAuth2 Token attack	403	Unauthorized	Error: Unauthorized
API Key attack	403	Unauthorized	Error: Unauthorized
Username attack	403	Unauthorized	Error: Unauthorized

The con  
the



### 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 in `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:

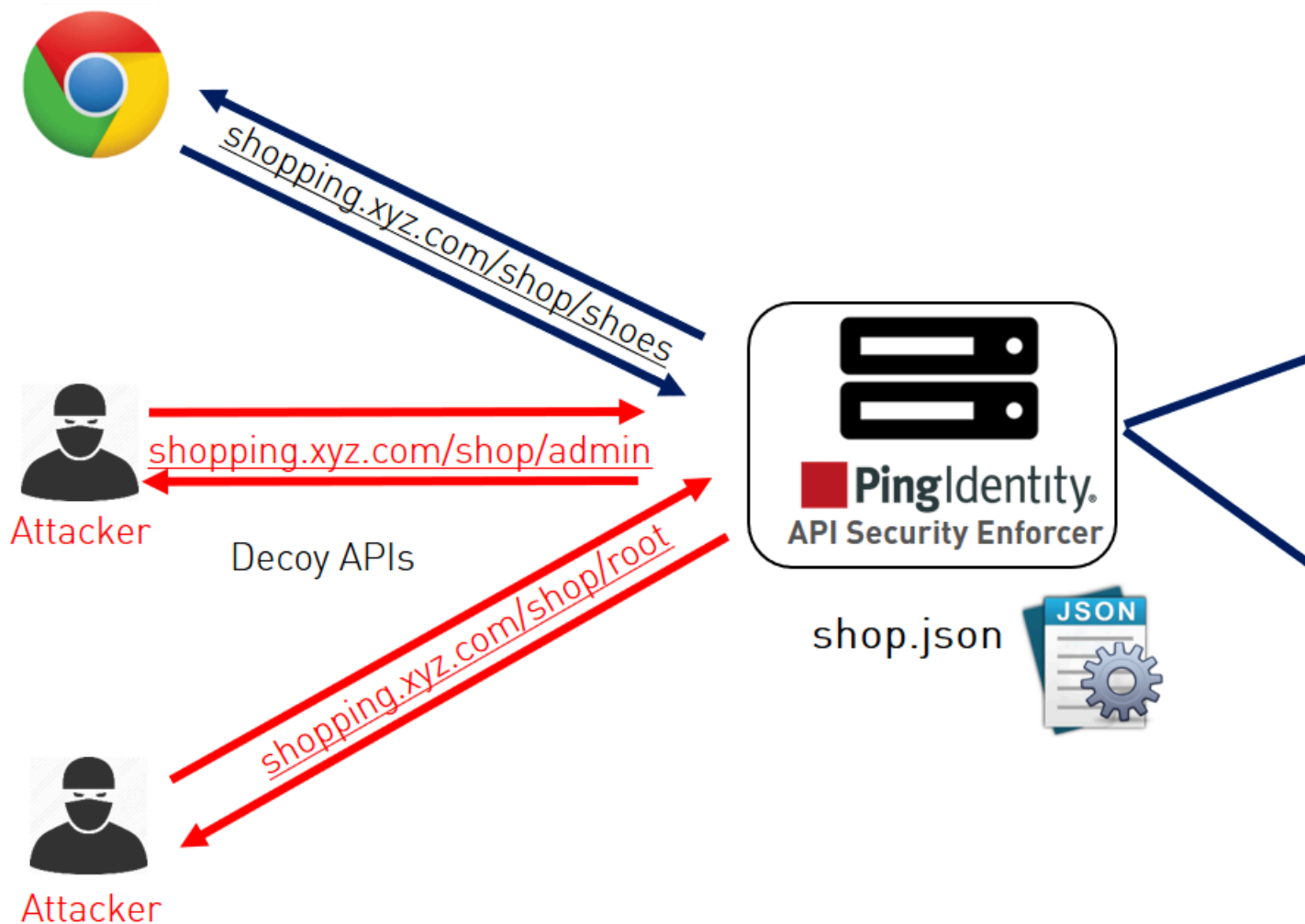
`decoy_pid_<pid_number>__yyyy-dd-mm-<log_file_rotation_time>` (for example, `decoy_pid_8787__2017-04-04_10-57.log`). decoy log files are rotated every 24-hours and stored in the `opt/pingidentity/ase/logs` directory.

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:

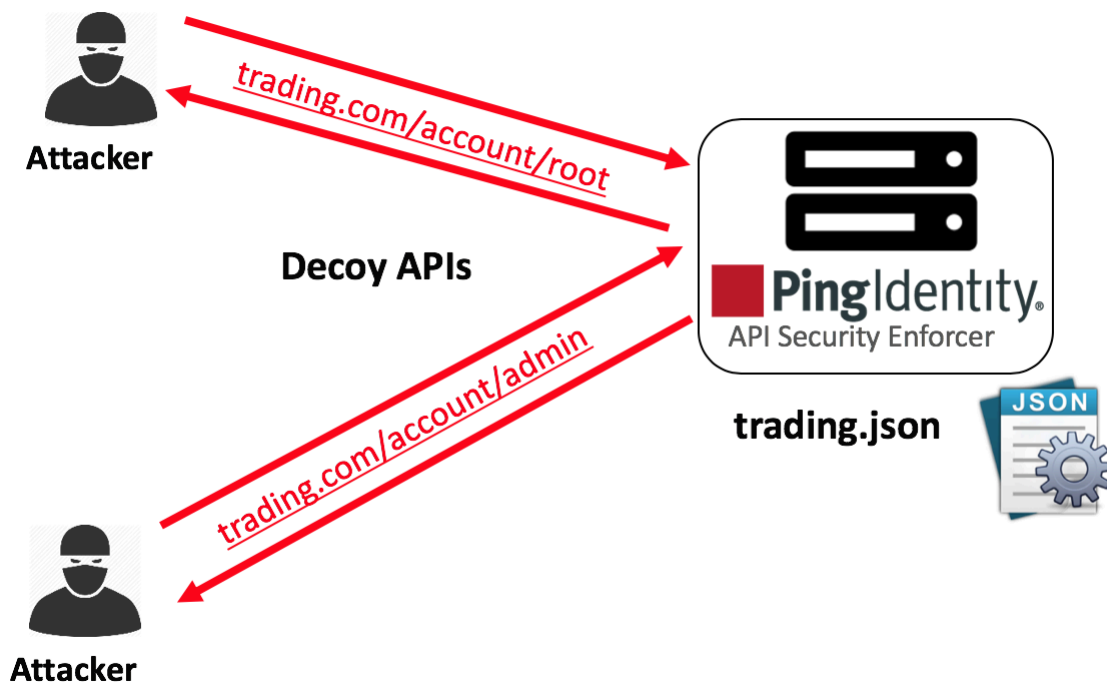
- decoy\_enabled parameter is set to true. If set to false, no decoy paths are configured.
- response\_code is set to 200. When a decoy sub-path is accessed, return a 200 response.
- response\_def is set to OK. When a decoy sub-path is accessed, return OK as the response.

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 WebSocket
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
	<b>Configured once in every API JSON file</b>
client_spike_threshold	Maximum requests per time-period from a single client IP to REST API. Time can be in seconds, minutes or hours.
server_connection_queueing	When <code>true</code> , queue API connection requests when all backends reach server connection quota. Default value is <code>false</code> .
	<b>Configured for each server in every API JSON file</b>

server_connection_quota	Maximum number of concurrent connections to a specific REST API on a server. Prevents aggregate connections from one or a cluster of ASE nodes from overloading a REST API running on a specific server.
server_spike_threshold	Maximum requests per time-period to the REST API running on a specified server. Prevents the aggregate request rate from one or a cluster of ASE nodes from overloading a REST API running on a specific server.  Time can be in seconds, minutes, or hours.

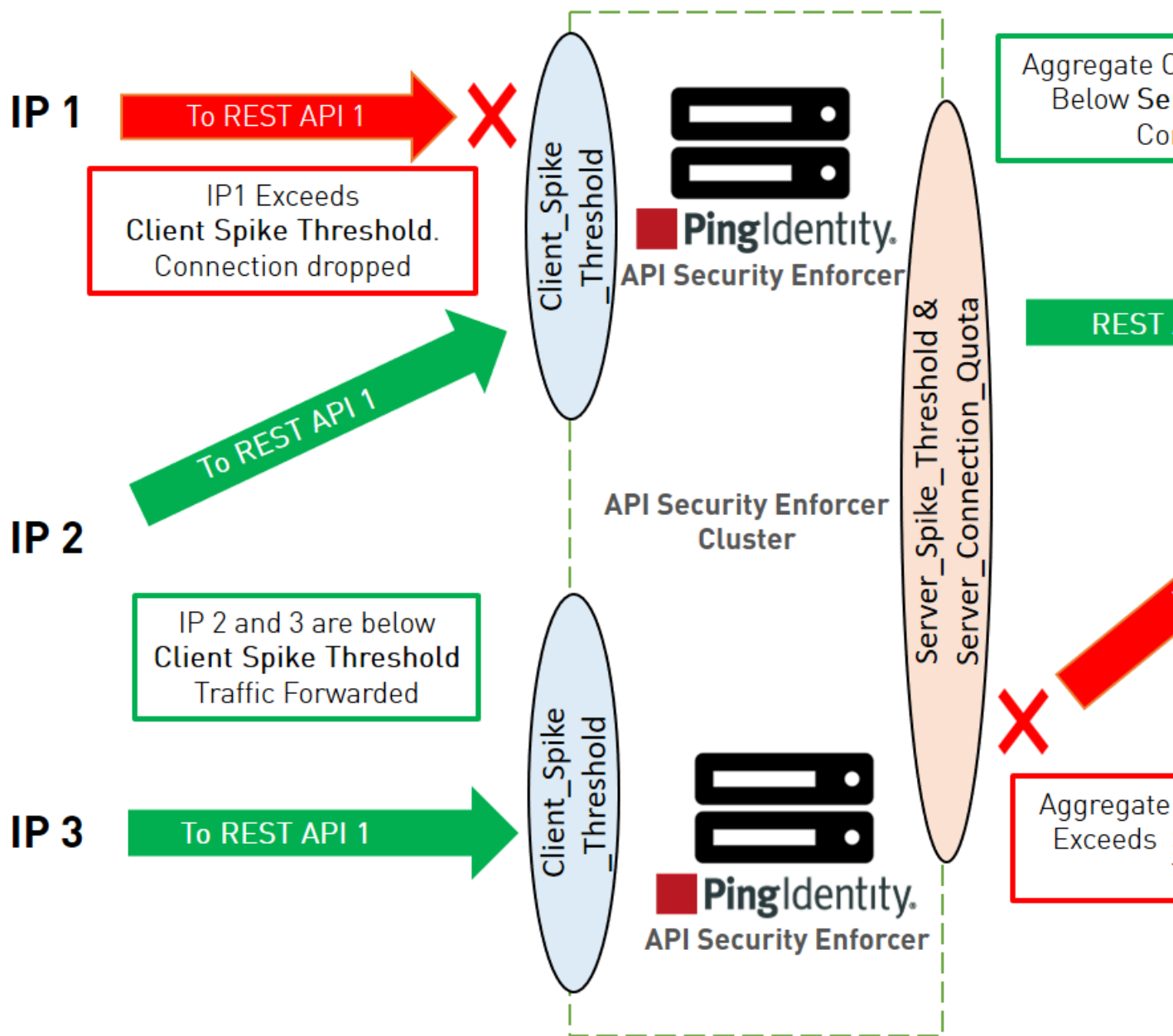
Client flow control monitors incoming traffic from each client connection and drops the session when traffic limits are exceeded. The diagram shows the following client scenarios:

- 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	30,000 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_queueing` is enabled. If queuing is not enabled, then the request is dropped.

### Summary table for REST API flow control

Parameter	Notes
<code>client_spike_threshold</code>	Maximum request rate from a client to an API
<code>server_spike_threshold</code>	Maximum aggregate request rate through ASE cluster nodes to a specific server.
<code>server_connection_quota</code>	Maximum number of concurrent sessions from ASE cluster nodes to a specific 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

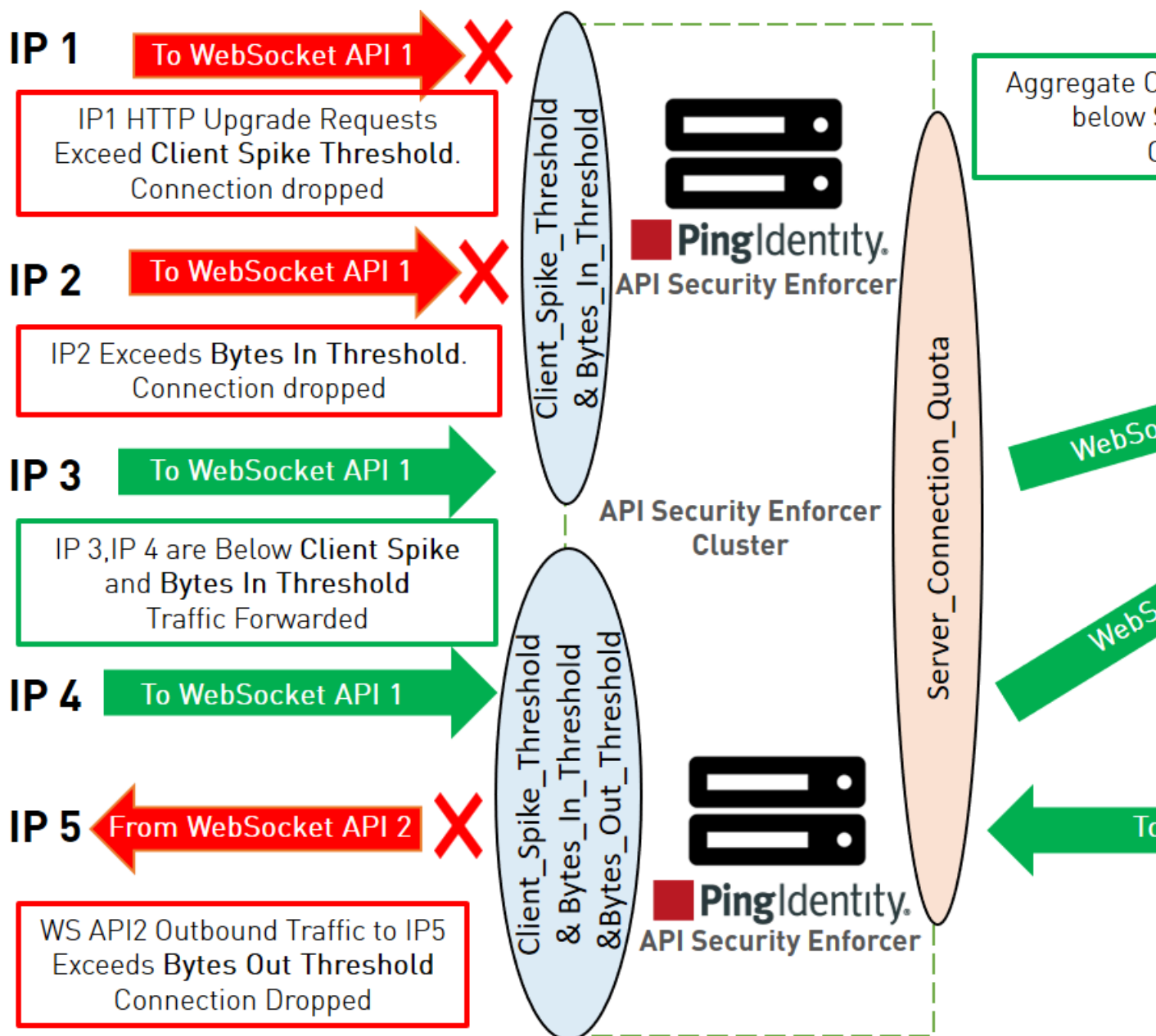
Flow control protects WebSocket servers using five control variables which are independently configured. By default, no flow control is enabled.

Variable	Description
<b>Configured once in every API JSON file</b>	
client_spike_threshold	<p>Maximum number of HTTP requests per time-period from a single IP to a specific WebSocket API.</p> <p>Time can be in seconds, minutes or hours.</p>
bytes_in_threshold	<p>Maximum number of bytes per time-period from a single IP to an ASE node.</p> <p>Time can be in seconds, minutes or hours.</p>
bytes_out_threshold	<p>Maximum number of bytes per time-period sent from an ASE node to a single IP.</p> <p>Time can be in seconds, minutes or hours.</p>

<code>server_connection_queueing</code>	When <code>true</code> , queue connection requests when all backend servers reach the server connection quota. The default value is <code>false</code> .
<b>Configured for each server in every API JSON file</b>	
<code>server_connection_quota</code>	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\_queueing 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 83 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 queuing 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:

$$\frac{Q_i}{\sum_{i=1}^n Q_i}$$

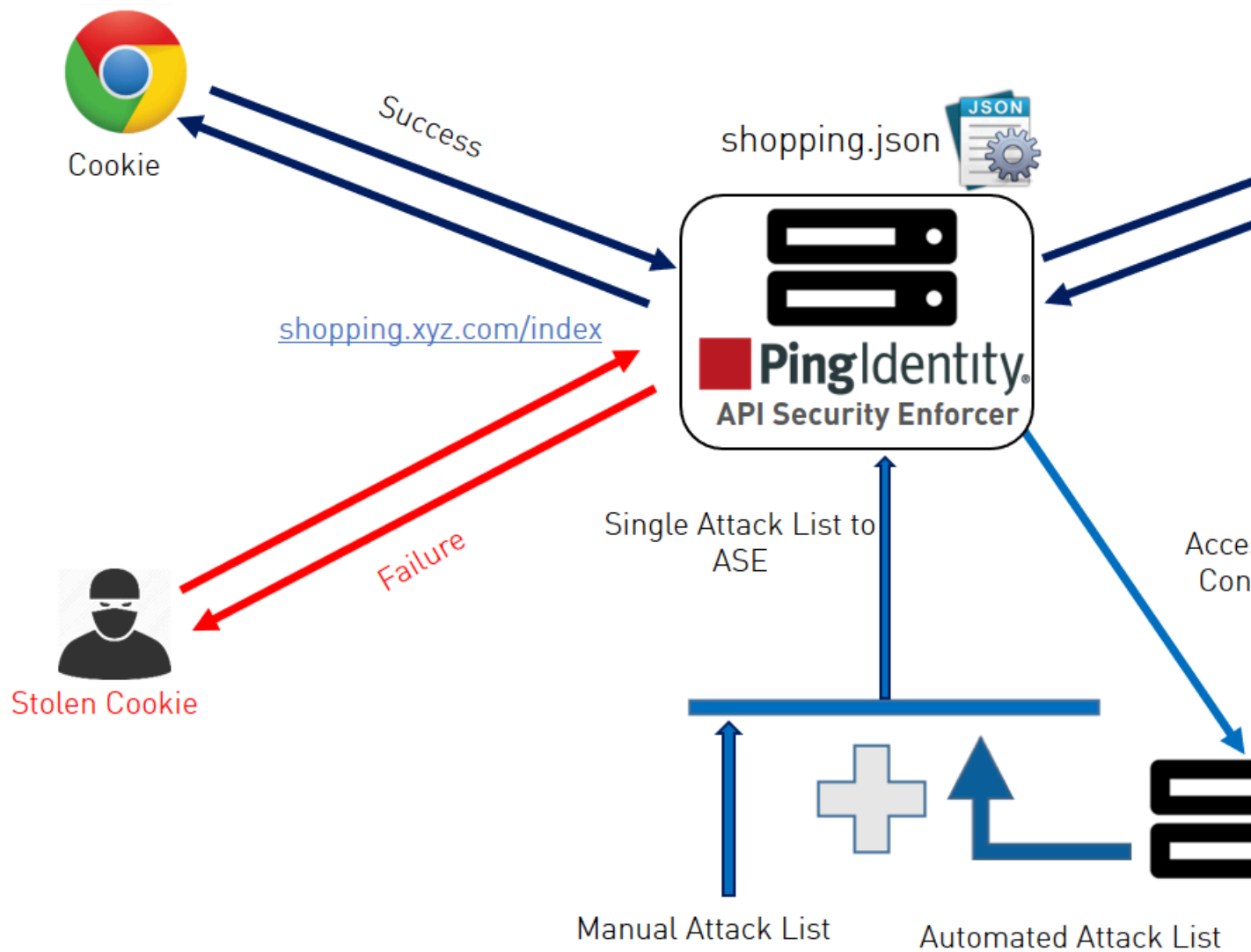
Where  $Q_i$  is the server connection quota for servers from  $i=1$  to  $i=n$

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
<code>abs_endpoint</code>	Hostname and port or the IPv4 and port of all the ABS nodes
<code>access_key</code>	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

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 MongoDB database. This value is obfuscated during the start of ASE.
enable_ssl	Set the value to true for SSL communication between ASE and ABS. The default value is true. ASE sends the access log files in plain text if the value is set to false.
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:jN6wflHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0
; 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 still 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, enter 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}
```

### 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_attacks
```

**Disable ASE Detected Attacks****Description**

Disable API firewall

**Syntax**

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

**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.



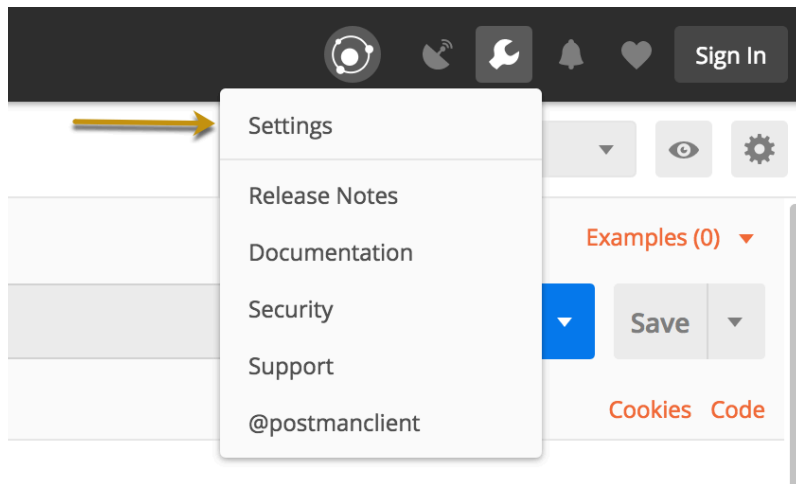
## ASE self-signed certificate with Postman

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:

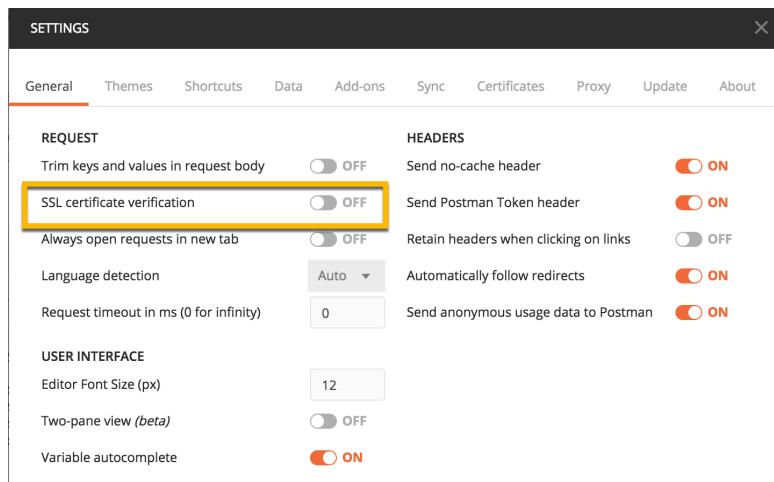
1.

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:

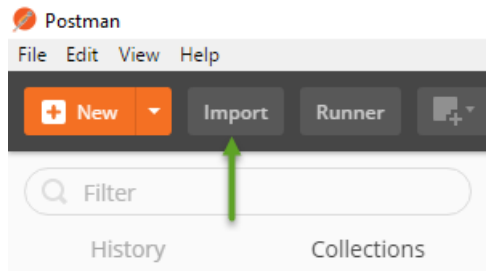


## View ASE REST APIs in Postman

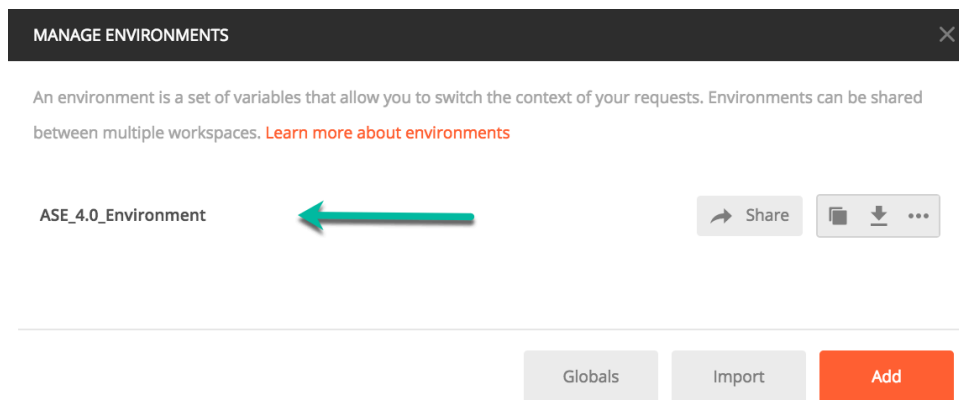
To view the reports, complete the following steps:

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`.

MANAGE ENVIRONMENTS
✕

Environment Name

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	⋮	Persist All	Reset All
<input checked="" type="checkbox"/>	ASE_IP	172.16.40.214	172.16.40.214			
<input checked="" type="checkbox"/>	Port	8010	8010			
<input checked="" type="checkbox"/>	Access_Key_Header	x-ase-access-key	x-ase-access-key			
<input checked="" type="checkbox"/>	Secret_Key_Header	x-ase-secret-key	x-ase-secret-key			
<input checked="" type="checkbox"/>	Access_Key	admin	admin			
<input checked="" type="checkbox"/>	Secret_key	admin	admin			
<input checked="" type="checkbox"/>	API_Name	pubatmapp	pubatmapp			
<input checked="" type="checkbox"/>	System_URL	https://{{ASE_IP}}:{{P...	https://{{ASE_IP}}:{{Port}}/v4/ase			
<input checked="" type="checkbox"/>	List_API	{{System_URL}}/api	{{System_URL}}/api			
<input checked="" type="checkbox"/>	API	{{List_API}}?api_id	{{List_API}}?api_id			
<input checked="" type="checkbox"/>	Server	{{System_URL}}/serv...	{{System_URL}}/server?api_id			
<input checked="" type="checkbox"/>	Cluster	{{System_URL}}/clust...	{{System_URL}}/cluster			
<input checked="" type="checkbox"/>	PersistentConnection	{{System_URL}}/pers...	{{System_URL}}/persistentconnection?api_id			
<input checked="" type="checkbox"/>	FireWall	{{System_URL}}/fire...	{{System_URL}}/firewall			
<input checked="" type="checkbox"/>	UpdateFireWall	{{FireWall}}?status	{{FireWall}}?status			
<input checked="" type="checkbox"/>	Blacklist	{{FireWall}}/blacklist...	{{FireWall}}/blacklist?tag			
<input checked="" type="checkbox"/>	Whitelist	{{FireWall}}/whitelist...	{{FireWall}}/whitelist?tag			
<input checked="" type="checkbox"/>	FlowControl	{{FireWall}}/flowcont...	{{FireWall}}/flowcontrol?api_id			
<input checked="" type="checkbox"/>	FlowControlPerServer	{{FireWall}}/flowcont...	{{FireWall}}/flowcontrol/server?api_id			
	Add a new variable					

ⓘ Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#) ✕

Cancel
Update

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
	<b>Note:</b> The default and only allowed access key is admin.
x-ase-secret-key	<Secret Key>
	<b>Note:</b> The default secret key is admin. You can change the default secret key using the <code>update_password</code> 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	<pre>{   "status": "success",   "status_message": "success" }</pre>
403	fail	<pre>{   "status": "api_already_exists",   "status_message": "api sample_api already exists" }</pre>
403	fail	<pre>{   "status": "validation_error",   "status_message": "&lt;detailed validation error description" }</pre>

### 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**

```
{
  "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	<pre>{ "status" : "success" , "status_message" :   "success" }</pre>
404	fail	<pre>{ "status" : "api_not_found" , "status_message" : "api sample_api does not exist" }</pre>

### 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	<pre>{ "status" : "success" , "status_message" :   "success" }</pre>
404	fail	<pre>{ "status" : "api_not_found" , "status_message" : "api sample_api does not exist" }</pre>

### 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	<pre>{ "status" : "success" , "status_message" : "success" }</pre>
404	fail	<pre>{ "status" : "api_not_found" , "status_message" : "api sample_api does not exist" }</pre>
403	fail	<pre>{ "status" : "validation_error" , "status_message" : "detailed info about validation error" }</pre>
403	fail	<pre>{ "status" : "server_exists" , "status_message" : "server already exists" }</pre>

### 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	<p><b>REST API</b></p> <pre>{   "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"   } ] }</pre> <p><b>WebSocket API</b></p> <pre>{   "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"   } ] }</pre>
404	fail	<pre>{"status" : "api_not_found" , "status_message" : "api sample_api does not exist"}</pre>

**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	<pre>{"status" : "success" , "status_message" :   "success" }</pre>
404	fail	<pre>{"status" : "api_not_found" , "status_message" : "api sample_api does not exist"}</pre>
404	fail	<pre>{"status" : "server_not_found" , "status_message" : "server does not exist"}</pre>
403	fail	<pre>{"status" : "validation_error" , "status_message" : "detailed info about json validation error"}</pre>

### 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	<pre>{   "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"     }   ] }</pre>
404	fail	<pre>{"status" : "no_cluster_mode" , "status_message" : "ase is not in cluster mode"}</pre>

### 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	<pre>{   "api_id" : "sample"   "persistent_connection_count" :   {     "ws":1,     "wss":0   } }</pre>
404	fail	<pre>{"status" : "api_not_found" , "status_message" : "api sample does not exist"}</pre>



**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	<p><b>If there is a status change</b></p> <pre>{   "status" : "enabled/disabled",   "status_message" : "Firewall is now enabled/disabled" }</pre> <p><b>If there is no change in status</b></p> <pre>{   "status" : "enabled/disabled",   "status_message" : "Firewall is already enabled/disabled" }</pre>
403	fail	<pre>{"status" : "invalid_value" , "status_message" : "query parameter status contains invalid value"}</pre>

### 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" : "ljkhasioutfdqbjfsdmakhflia"
}
```

#### Response

Status code	Response body
200 OK	Cookie JSESSIONID ljkhasioutfdqbjfsdmakhflia 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)
	<ul style="list-style-type: none"> <li>▪ all</li> <li>▪ manual</li> <li>▪ abs_generated</li> <li>▪ invalid_content_type</li> <li>▪ invalid_method</li> <li>▪ invalid_protocol</li> <li>▪ decoy</li> </ul>
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 ljkhasioutfdqbjbfdmakhflia 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" : "ljkhasioutfdqbjjsfdmakhflia"
}

```

## Response

Status code	Response body
200 OK	Cookie JSESSIONID ljkhasioutfdqbjjsfdmakhflia added to whitelist
403 Forbidden	Cookie JSESSIONID ljkhasioutfdqbjjsfdmakhflia 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" : "ljkhasioutfdqbjjsfdmakhflia"
}

```

**Response**

Status code	Response body
200 OK	Cookie JSESSIONID ljkhasioutfdqbjjsfdmakhflia added to whitelist
403 Forbidden	Cookie JSESSIONID ljkhasioutfdqbjjsfdmakhflia 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	<p><b>Flow control for REST API</b></p> <pre>{   "api_id": "api_name"   "flow_control": {     "client_spike_threshold": "0/second",     "server_connection_queueing": false   } }</pre> <p><b>Flow control for WebSocket API</b></p> <pre>{   "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   } }</pre>
403	fail	<pre>{"status" : "validation_error" ,   "status_message" : "&lt;detailed                     validation error description" }</pre>
404	fail	<pre>{"status" : "api_not_found" , "status_message" : "api sample does not                     exist"}</pre>

### 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	<p><b>Flow control for REST APIs</b></p> <pre>{   "api_id": "api_name"   "flow_control": {     "client_spike_threshold": "0/second",     "server_connection_queueing": false   } } </pre> <p><b>Flow control for WebSocket APIs</b></p> <pre>{   "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   } } </pre>
403	fail	<pre>{ "status" : "validation_error" ,   "status_message" : "&lt;detailed                         validation error description" } </pre>
404	fail	<pre>{ "status" : "api_not_found" , "status_message" : "api sample does not                         exist" } </pre>

## 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	<pre> {   "status": "success",   "status_message": "server updated successfully" } </pre>
403	fail	<pre> {"status" : "validation_error" ,  "status_message" : "&lt;detailed                     validation error description" } </pre>
404	fail	<pre> {"status" : "api_not_found" , "status_message" : "api sample does not                     exist"} </pre>

## 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

Create role 1 2 3

Select type of trusted entity

**AWS service**  
EC2, Lambda and others

**Another AWS account**  
Belonging to you or 3rd party

**Web identity**  
Cognito or any OpenID provider

**SAML 2.0 federation**  
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**Lambda**  
Allows Lambda functions to call AWS services on your behalf.

API Gateway	DMS	Elastic Transcoder	Machine Learning	SageMaker
Application Auto Scaling	Data Pipeline	ElasticLoadBalancing	MediaConvert	Service Catalog
Auto Scaling	DeepLens	Glue	OpsWorks	Step Functions
Batch	Directory Service	Greengrass	RDS	Storage Gateway
CloudFormation	DynamoDB	GuardDuty	Redshift	
CloudHSM	EC2	Inspector	Rekognition	
CloudWatch Events	EMR	IoT	S3	
CodeBuild	ElastiCache	Kinesis	SMS	
CodeDeploy	Elastic Beanstalk	Lambda	SNS	
Config	Elastic Container Service	Lex	SWF	

\* Required Cancel Next: Permissions

instance:

2. Assign **AmazonEC2ReadOnlyAccess** privilege to the

Create role 1 2 3

Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy Refresh

Filter: Policy type  Showing 367 results

<input type="checkbox"/>	Policy name	Attachments	Description
<input type="checkbox"/>	AmazonEC2ContainerServiceAutoscaleRole	0	Policy to enable Task Autoscaling for Amazon EC2 Contai...
<input type="checkbox"/>	AmazonEC2ContainerServiceEventsRole	0	Policy to enable CloudWatch Events for EC2 Container Se...
<input type="checkbox"/>	AmazonEC2ContainerServiceforEC2Role	0	Default policy for the Amazon EC2 Role for Amazon EC2 ...
<input type="checkbox"/>	AmazonEC2ContainerServiceFullAccess	0	Provides administrative access to Amazon ECS resources.
<input type="checkbox"/>	AmazonEC2ContainerServiceRole	0	Default policy for Amazon ECS service role.
<input type="checkbox"/>	AmazonEC2FullAccess	2	Provides full access to Amazon EC2 via the AWS Manage...
<input checked="" type="checkbox"/>	AmazonEC2ReadOnlyAccess	5	Provides read only access to Amazon EC2 via the AWS M...
<input type="checkbox"/>	AmazonEC2ReportsAccess	0	Provides full access to all Amazon EC2 reports via the AW...
<input type="checkbox"/>	AmazonEC2RoleforAWSCodeDeploy	0	Provides EC2 access to S3 bucket to download revision. T...
<input type="checkbox"/>	AmazonEC2RoleforDataPipelineRole	0	Default policy for the Amazon EC2 Role for Data Pipeline s...
<input type="checkbox"/>	AmazonEC2RoleforSSM	0	Default policy for Amazon EC2 Role for Simple Systems M...
<input type="checkbox"/>	AmazonEC2SpotFleetAutoscaleRole	0	Policy to enable Autoscaling for Amazon EC2 Spot Fleet

\* Required Cancel Previous Next: Review

role.

3.

**Create role**

1 2 3

**Review**

Provide the required information below and review this role before you create it.

**Role name\***   
Use alphanumeric and '+,=,@,-' characters. Maximum 64 characters.

**Role description**   
Maximum 1000 characters. Use alphanumeric and '+,=,@,-' characters.

**Trusted entities** AWS service: ec2.amazonaws.com

**Policies** AmazonEC2ReadOnlyAccess [↗](#)

Provide the role name: \* Required Cancel Previous **Create role**

## 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` files. Note that `balancer.log` file is not rotated while `controller.log` file is rotated every 24-hours. For more information on ASE logs, see [ASE management, access and audit logs](#) on page 32

**controller.log** messages:

Log message	Description
unknown cluster uuid	This message is logged in <code>controller.log</code> when a ASE node with a different cluster ID or secret key tries to join an ASE cluster. For more information, see <a href="#">Start ASE cluster</a> on page 22
resolve error	This message is logged in <code>controller.log</code> when ASE is not able to resolve ABS or server hostname
connect error	This message is logged in <code>controller.log</code> when ASE is not able to connect to ABS or a server.
handshake error	This message is logged in <code>controller.log</code> when connection to ABS or server because of problems in SSL handshake.
error while sending message to lb connection	This message is logged in <code>controller.log</code> 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 <code>controller.log</code> when there is a IPC connection failure between ASE's controller and balancer modules
License file <i>&lt;license file path&gt;</i> is expired. Please renew your license	This message is logged in <code>controller.log</code> when PingIntelligence license has expired. For more information, see <a href="#">ASE license</a> on page 13.
Unexpected Error	This message is logged in <code>controller.log</code> when ASE's controller module is unavailable. This is a fatal error.
info   event   event type : <i>&lt;event type&gt;</i> event value : <i>&lt;value of event&gt;</i>	The following events are logged logs even if email alert is not enabled: <ul style="list-style-type: none"> <li>▪ Cluster node up</li> <li>▪ Cluster node down</li> <li>▪ server state changed to Up</li> <li>▪ server state changed to Down</li> <li>▪ log upload service failed</li> <li>▪ error while uploading log file</li> </ul> If email_alert is enabled, then all events will be available in logs. Fore more information, see <a href="#">Email alerts and reports</a> on page 35
api memory limit reached. total number of cookies dropped <i>&lt;count&gt;</i>	This message is logged in <code>controller.log</code> when ASE is dropping cookies because of low API memory. For more information, see <code>api_memory_size</code> in <a href="#">Defining an API – API JSON configuration file</a> on page 50
stopping API Security Enforcer	This message is logged in <code>controller.log</code> when ASE stops.
API Security Enforcer started	This message is logged in <code>controller.log</code> 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 : <i>process_id_number</i>   tcp_fe   !!!! low memory : connection dropped due to low memory !!!!	This message is logged in <code>controller.log</code> when ASE is runnig low on memory because of which ASE drops the client connections.

# ABS AI Engine

---

## Features at a glance

---

API Behavioral Security Engine (ABS) v4.0 delivers the following features:

- Deeper insight into API activity:
  - User activity reports with information on APIs accessed, tokens/IPs used, URLs accessed, and other details.
  - Forensic reporting on a specific user or API key activity including actions leading up to an attack.
- Enhanced AI-based attack detection including:
  - Account takeover based on user behavior accessing APIs, the sequence of APIs accessed, and aggregate API activity including extraction or injection of data.
  - API key attack detection includes detection of stolen API keys and anomalous behavior.
  - User names and API keys can be automatically added to the blacklist and blocked from API access. Note: user context requires a V4.0 sideband interface supporting user variables.
- Updated PingIntelligence Dashboard with improved navigation and black list entries.
- Improved administration tools include:
  - Attack management tool for unblocking clients and adjusting AI engine thresholds
  - User controlled settings for retention of black list entries by client identifier type – for example, block IP addresses for 7-days, block tokens for 1-day.
  - Enhanced syslog reporting including integration with Splunk and other SIEM platforms.

### ABS features

- Comprehensive reporting on API traffic including:
  - API activity by token, cookie, or API key - client information, API and URL usage, metrics, anomalies, backend errors, attack types, and decoy traffic. Reporting provides information for compliance reports.
  - Forensic reporting on activity from an individual IP address, token, or cookie. Understand activity leading up to an attack.
  - API deception reporting – information on hackers blocked for access of Decoy APIs.
  - Token-based metrics and forensics reporting – reports which show token activity on an API and detailed reporting of all activity for a specific token.
  - REST API interface for access to API JSON reports which are also accessible by customer management systems for API monitoring.
- Advanced Artificial Intelligence and machine learning software detects suspected cyber attacks including complex attacks requiring correlation across multiple clients.
  - Attack examples include data exfiltration, stolen token attacks, content scraping, cross-API attacks, extended time frame attacks (e.g. slow login, extended data exfiltration), API Distributed Denial of Service (DDoS) attacks, API memory attacks, and brute force login attacks.
  - Automatic sending of client blacklists to API Security Enforcer (ASE) for use in terminating sessions and blocking future access for rogue clients.
- API discovery – Option to automatically discover APIs in your ecosystem.
  - Automated API Definition (AAD) converts discovered APIs to API Security Enforcer (ASE) compatible API JSON files and automatically adds the files to ASE.
- PingIntelligence for APIs Dashboard (installed separately) uses Kibana graphs to provide API summary information on metrics, anomalies, and attacks for each API.
  - Discovering and reporting on valid and invalid URLs used to access each secured API.

- Stateless ABS clustering with redundant MongoDB databases for scale and high availability.
- Automatic generation of alerts to notify operations of errors or capacity issues.
- Flexible deployment options on standard Linux servers (RHEL 7.6, Ubuntu 16.04 LTS) running JVMs across various environments such as Containers, VMs, bare metal servers, and Clouds.
- Sensitive data obfuscation
  - Obfuscation of keys and passwords in the properties file ABS.
  - CLI to generate master key and obfuscate keys and passwords.
  - Restricted user for ABS reports and Dashboard.
  - SSL communication between ABS and ASE as well as for ABS REST APIs
- Script to update training period, system threshold interval, discovery interval and enabling and disabling discovery.

## 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

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 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.

### 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

The transaction count is updated every day after 00:00 hours midnight in the `/opt/pingidentity/abs/logs/abs.log` file. To check the current monthly transaction count, `grep` for `Current Transactions` in the `abs.log` file. Following is a sample snippet for the current number of transactions:

```
$ grep "Current Transactions" *
abs.log:2018-12-19 00:01:25 INFO - Current Transactions: 289088158 between
earlier date: Sat Dec 01 00:00:00 EST 2018 and later date: Tue Dec 18
23:59:59 EST 2018
```

The `earlier date` is always the starting date of the month.

## 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 4.0.1...
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 machine 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
```

## 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, 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>" } )
```

## 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
<code>config</code>	Contains <code>abs.properties</code> , a Java properties file used to configure ABS.
<code>data</code>	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.  <b>Note:</b> Do not edit the scripts in the <code>bin</code> directory.
mongo	Contains the <code>abs_init.js</code> file used to load the default schema, secret key, and access key.
util	Contains utilities to: <ul style="list-style-type: none"> <li>▪ Check and Open MongoDB Default Port</li> <li>▪ Purge the Processed Access Logs from ABS</li> <li>▪ Purge ABS Data from MongoDB</li> <li>▪ <code>open_ports_abs.sh</code>: Open the default ports 8080 and 9090 for ABS REST API and connectivity from ASE respectively. Run the script on the ABS machine.</li> </ul>

## Obfuscate passwords

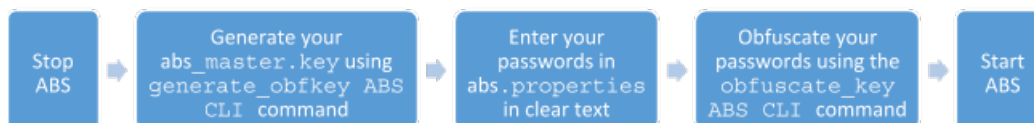
Using ABS command line interface, you can obfuscate the keys and passwords configured in `abs.properties`. The keys and passwords obfuscated include:

- `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.

## 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_certificate.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 master and slave. 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
<b>ABS IP, port, log level, and JKS password</b>	
<code>host_ip</code>	The externally visible IP address of the host ABS machine.
<code>management_port</code>	Port for ABS to ASE and REST API to ABS communication. The default value is 8080.
<code>log_port</code>	Port for ASE to send log files to ABS. The default value is 9090.
<code>jks_password</code>	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 <a href="#">importing your CA-signed certificate</a> .
<code>log_level</code>	Log detail captured. The default is INFO. Additional options - DEBUG, ERROR.
<b>ABS performance configurations</b>	
<code>system_memory</code>	Memory size in MB allocated to run machine learning jobs. Recommended to be at least 50% of system memory.
<code>system_json_size</code>	Memory size in KB allocated for API JSON files. The default is 500 KB.
<code>runaway_time</code>	Maximum time in minutes to wait for a job to finish. The default value is 120 minutes.
<code>queue_size</code>	Do not change the value of this parameter. The default is 10.
<b>ABS email configurations for alerts and reporting</b>	

<code>enable_emails</code>	Enable ( <code>true</code> ) or disable ( <code>false</code> ) ABS email notifications.
<code>sender_email</code>	Email address used for sending email alerts and reports.
<code>receiver_email</code>	Email address notified about alerts and reports. If you want more than one person to be notified, use an email alias.
<code>email_password</code>	Password of sender's email account.
	<p><b>Note:</b> You can leave this field blank if your SMTP server does not require authentication.</p>
<code>smtp_port</code>	Port number of SMTP server.
<code>smtp_host</code>	Hostname of SMTP server.
<code>smtp_ssl</code>	<p>Set to <code>true</code> if you want email communication to be over SSL. Make sure that the SMTP server supports SSL. If you set <code>smtp_ssl</code> to <code>true</code> and the SMTP server does not support SSL, email communication falls back to the non-SSL channel. The default value is <code>true</code>.</p> <p>Set it to <code>false</code> if email communication is over a non-SSL channel. The email communication will fail if you set the parameter to <code>false</code>, but the SMTP server only supports SSL communication.</p>
<code>smtp_cert_verification</code>	<p>Set to <code>true</code> if you want ABS to verify the SMTP server's SSL certificate. The default value is <code>false</code>.</p> <p>If you set it to <code>false</code>, ASE does not verify SMTP server's SSL certificate; however, the communication is still over SSL.</p> <p><b>Note:</b> If you have configured an IP address as <code>smtp_host</code> and set <code>smtp_cert_verification</code> to <code>true</code>, then make sure that the certificate configured on the SMTP server has the following:</p> <pre>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</pre> <p>Here <code>x.x.x.x</code> is the IP address is the address configured in <code>smtp_host</code>.</p>
<b>MongoDB configuration</b>	
<code>mongo_rs</code>	Comma separated MongoDB replica set nodes IP addresses and port numbers. A maximum of three nodes can be configured.
<code>metadata_dbname</code>	<p>The MongoDB metadata database name.</p> <p>The default value is <code>abs_metadata</code>.</p>

data_dbname	The MongoDB data database name. The default value is <code>abs_data</code> .
mldata_dbname	The MongoDB machine learning database name. The default value is <code>abs_mldata</code>
mongo_username	Username of MongoDB.
<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <i>i</i> <b>Note:</b> Required for MongoDB authentication         </div>	
mongo_password	MongoDB password
mongo_ssl	Set it to <code>true</code> if MongoDB is configured to use SSL connections. The default value is <code>false</code> .
<b>ABS reporting node</b>	
dashboard_node	When <code>true</code> , 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 <code>false</code> .
<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <i>i</i> <b>Note:</b> Multiple nodes can be Reporting or Dashboard nodes.         </div>	

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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCu
# 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 you don't have authentication enabled in MongoDB, leave blank following
# two keys
mongo_username=absuser
mongo_password=OBF:AES:Q3vcrnj7VZILTPdJnxkOsyimHRvGDQ==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFC
# 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
```



```

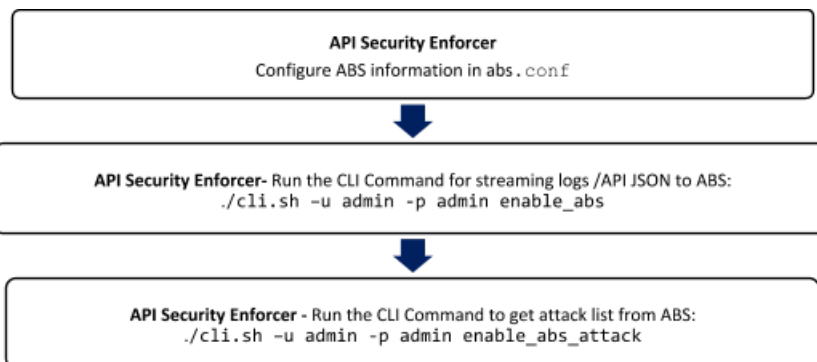
# Time to mark a job runaway in minutes
runaway_time=120
# Job queue size per node
queue_size=10
# Setting as true makes an ABS node for dashboard query only and does not
# participate in ABS cluster for log processing
dashboard_node=false
# Memory for webserver and streaming server (unit is in MB)
system_memory=4096
# 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:~N6wflLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0
; secret key for abs node

```

```
secret_key=OBF:AES:Y2DadCU4JFZp3bx8Ehn0iw: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).

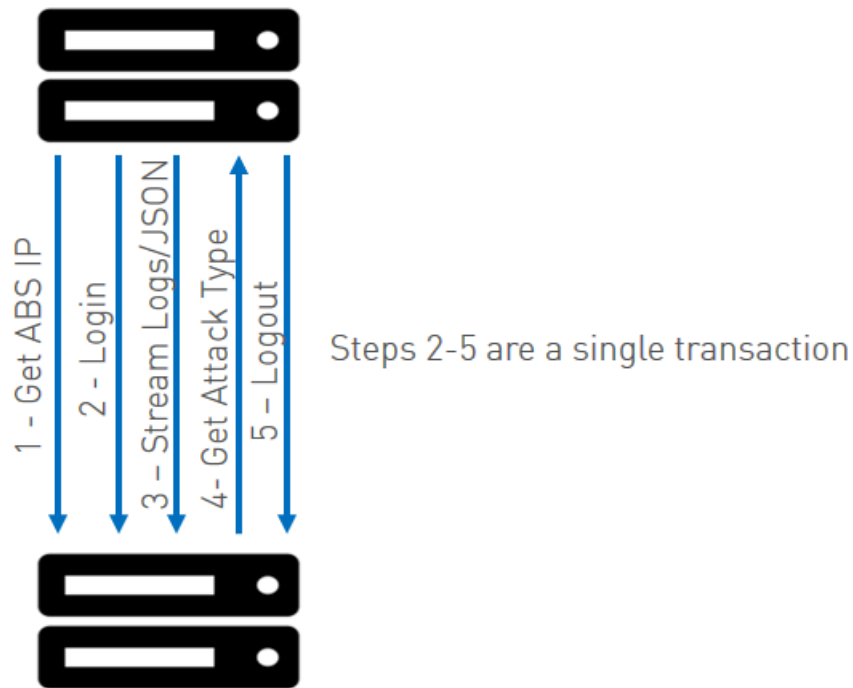
ABS uses machine learning (ML) algorithms to discover attacks, anomalies, and other traffic information. It stores incoming API Security Enforcer logs and then passes these logs to the machine learning engine for analysis. In high load environments, a single ABS node may not be able to process all log 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.

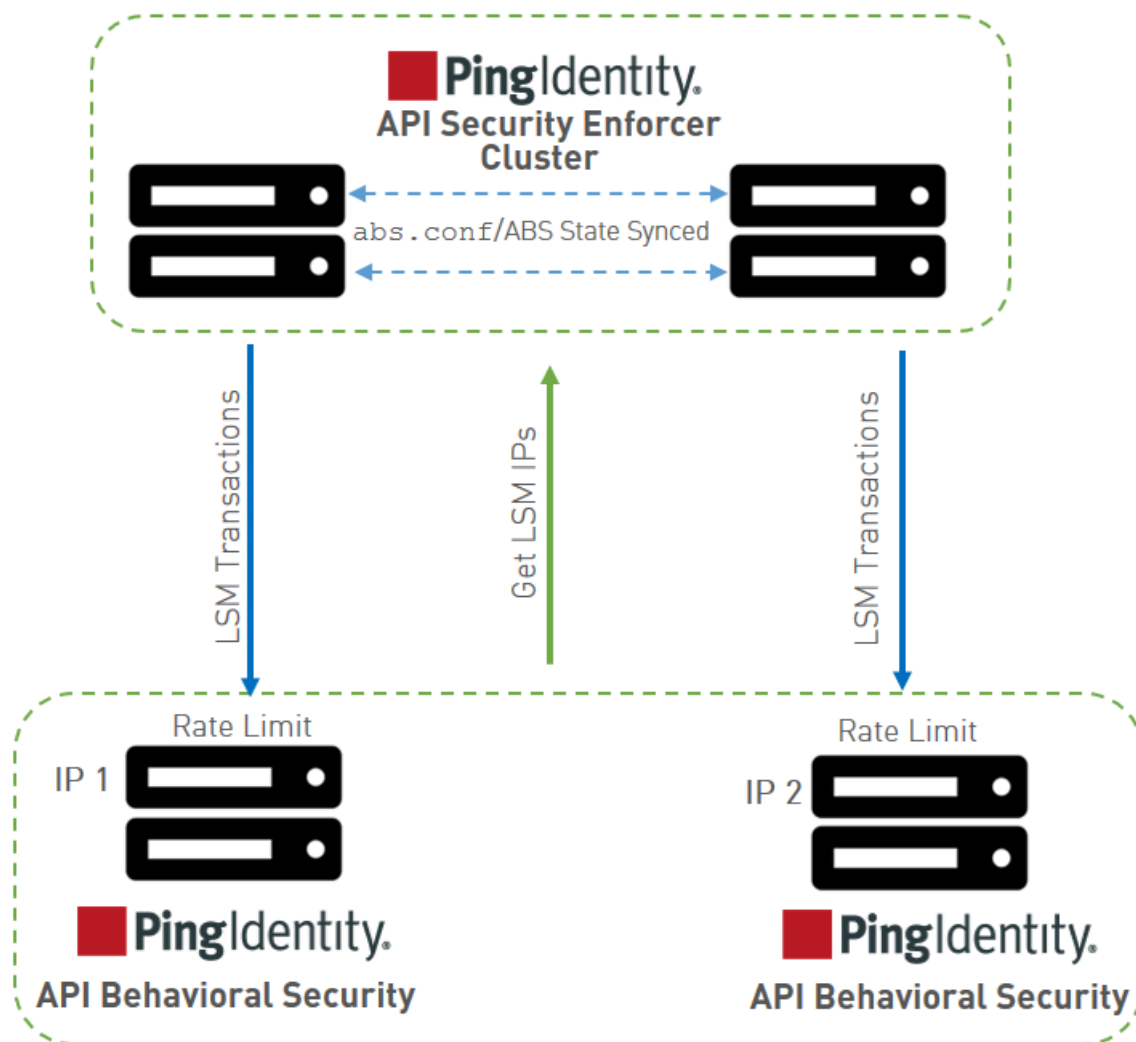
## PingIdentity. API Security Enforcer



## PingIdentity. API Behavioral Security

### 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.

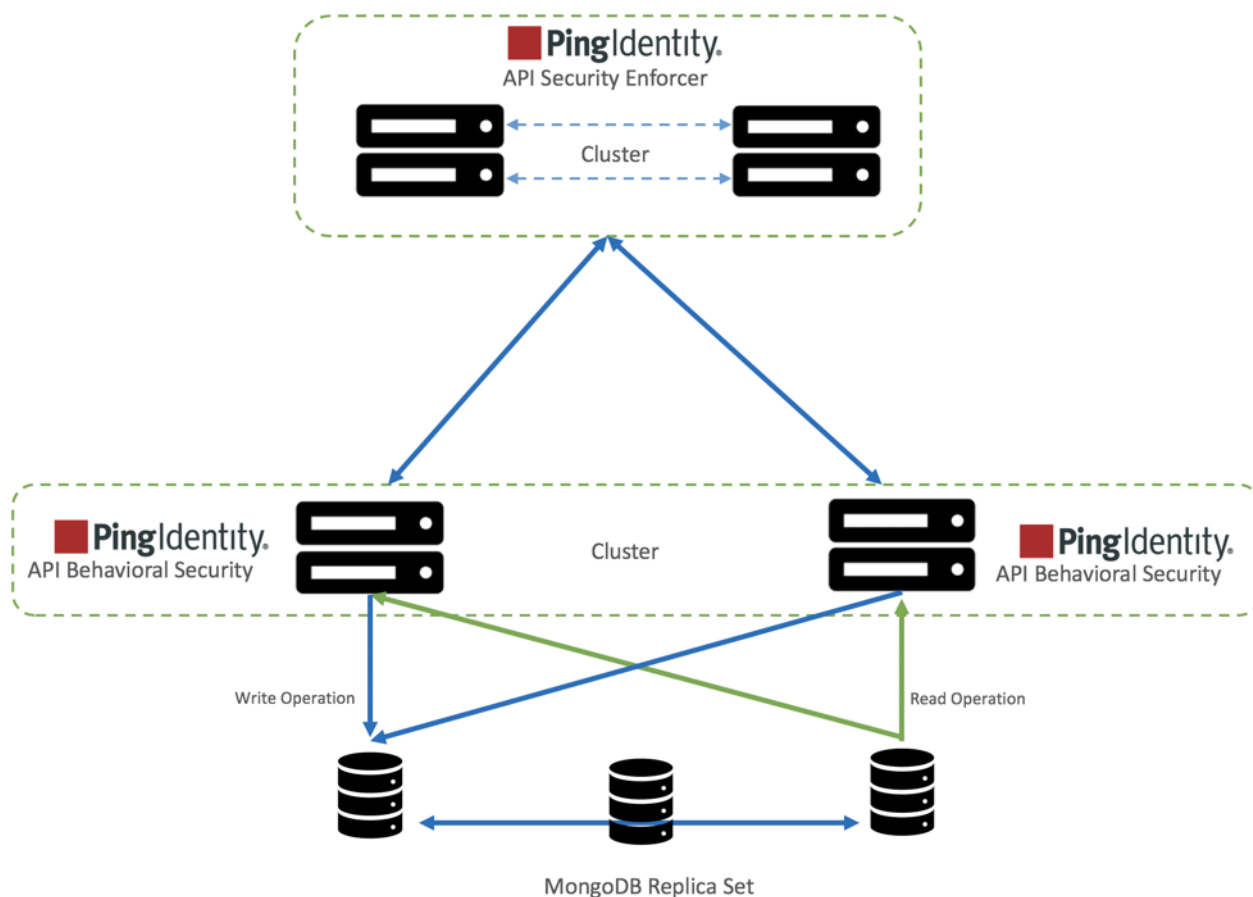


## ABS cluster

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.<yyy>-<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_21/9k2unv5l2bsgurteot3s3pmt03/ : last changed at Mon Jan
10 11:32:31 IST 2018
removing /opt/ pingidentity/abs/data/2018-04-10-11_21/
ilq67a3g5sve2pmpkcp271o37c/ : 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 SSL, use the `--ssl` option. The following examples assumes that MongoDB is configured to use 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 --ssl --
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 --ssl --
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 --ssl --
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 --ssl --
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 SSL.

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 --ssl --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 --ssl --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 --ssl --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 --ssl
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: mongod://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
2019-10-11T05:46:43.802+0000 W NETWORK [js] SSL peer certificate validation failed: self signed certificate
Implicit session: session { "id" : UUID("400fcaa5-57dd-4123-a5e6-b54cle0bdfda") }
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
      }
    ]
  }
}

```

## 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/pingidentity/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 occurred>
```

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	<p><b>Cause:</b> A possible reason for this alert could be that historical access log files from ASE have accumulated on the storage disk.</p> <p><b>Resolution:</b> Purge or archive the old access log files from storage disk.</p>

ABS node added to cluster	<p>ABS sends an email alert when a node joins an ABS cluster.</p> <p><b>Confirm:</b> ABS admin should verify whether the correct ABS node joined the ABS cluster.</p>
ABS node removed from cluster	<p>ABS sends an email alert when a node is removed from an ABS cluster.</p> <p><b>Confirm:</b> 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.</p>
Memory scale up or scale down	<p><b>Cause:</b> 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.</p> <p><b>Resolution:</b> 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.</p>
CPU scale up or scale down	<p><b>Cause:</b> 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.</p> <p><b>Resolution:</b> 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.</p>
Disk scale up or scale down	<p><b>Cause:</b> 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.</p> <p><b>Resolution:</b> 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.</p>
License <path> is invalid. ABS will shut down now	<p><b>Cause:</b> 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.</p> <p><b>Resolution:</b> Validate current license file path. Also check for file permissions of the license file.</p>
ABS license at <path> has expired. Please renew your license.	<p><b>Cause:</b> ABS sends this email alert when ABS license has expired. The license expires at the end of the license period.</p> <p><b>Resolution:</b> Renew your ABS license.</p>
Maximum transaction limit reached for the current month	<p>ABS sends this warning message when ABS crosses the licensed monthly transaction limit.</p>

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	<p><b>Cause:</b> 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.</p> <p><b>Resolution:</b> Check MongoDB Primary node status to bring it back online or add additional secondary node if needed.</p>

## 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

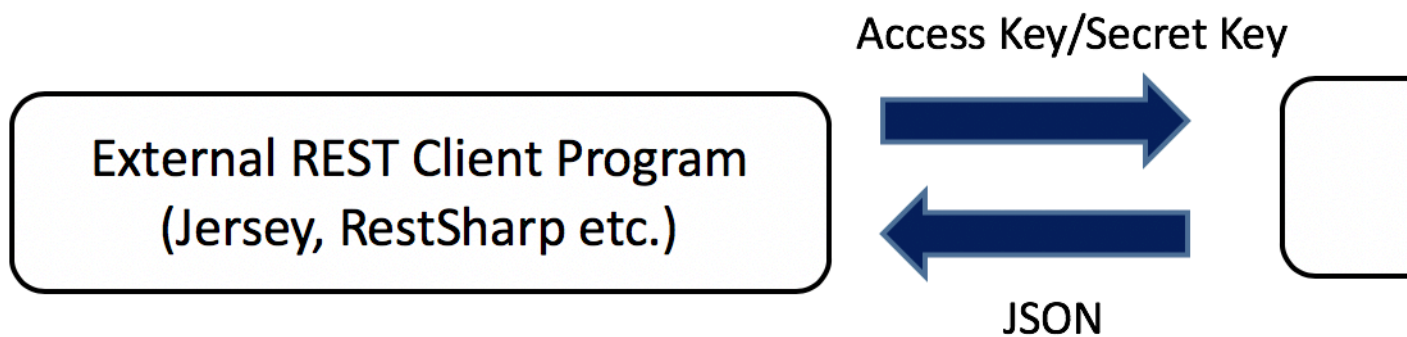
ABS provides external REST APIs which are used to access JSON reports providing deep insight into the following:

- 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 <code>earlier_date</code> 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 <code>later_date</code> 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:

- **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>

Here is the JSON report.

```
{
  "company": "ping identity",
  "name": "api_admin",
  "description": "This report contains status information on all APIs, ABS
clusters, and ASE logs",
  "across_api_prediction_mode": true,
  "api_discovery": {
    "subpath_length": "1",
    "status": true
  },
  "apis": [
    {
      "api_name": "apiKeypubatmapp",
      "host_name": "*",
      "url": "/",
      "api_type": "regular",
      "creation_date": "Thu Oct 10 10:31:01 UTC 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": "",
      "token": true,
      "training_started_at": "n/a",
      "training_duration": "n/a",
      "prediction_mode": false,
      "apikey_header": "",
      "apikey_qs": "QS_API_KEY"
    },
    {
      "api_name": "ws",
      "host_name": "*",
      "url": "/app",
      "api_type": "decoy-incontext",
      "creation_date": "Thu Oct 10 10:31:01 UTC 2019",
      "servers": 1,
      "protocol": "ws",
      "cookie": "JSESSIONID",
      "token": false,
      "training_started_at": "Thu Oct 10 10:32:53 UTC 2019",
      "training_duration": "1 hour",
    }
  ]
}
```



```

        "prediction_mode": true,
        "apikey_header": "",
        "apikey_qs": ""
    }
],
"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"
        }
    ]
},
"ase_logs": [
    {
        "ase_node": "317ea1e4-4f52-4784-9c3a-80f659ac6162",
        "last_connected": "Thu Oct 10 15:11:05 UTC 2019",
        "logs": {
            "start_time": "Thu Oct 10 10:31:06 UTC 2019",
            "end_time": "Thu Oct 10 15:11:05 UTC 2019",
            "gzip_size": "80.11MB"
        }
    }
],
"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
    }
]
}
}

```

**Percentage diskusage limit:** The percentage disk usage limit is configured in the `/pingidentity/abs/config/abs.init` 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

```

You need to restart all the ABS node for your changes to take effect.

### 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

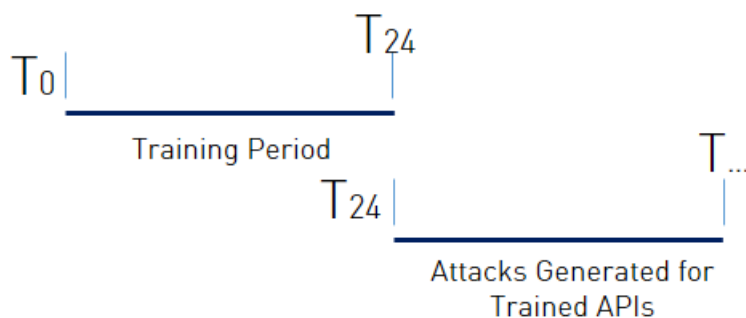
The ABS AI Engine needs to be trained before it can detect attacks on API services. The AI engine training is governed by global variables which are configured in the `/opt/pingidentity/abs/mongo/abs_init.js` file. The AI training runs for the minimum training time set in the `abs_init.js` file but a 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
<code>attack_initial_training</code>	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.
<code>attack_update_interval</code>	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 <code>continuous_learning</code> is set to <code>true</code> .
<code>continuous_learning</code>	Setting this value to <code>true</code> configures the AI model to learn continuously based on the live traffic. If it is set to <code>false</code> , the AI model detects attack based on the initial training.
<code>window_length</code>	The maximum time period that the AI model uses to detect attacks across APIs. The default and maximum value for <code>window_length</code> is 24-hours. The training period should be longer than the <code>window_length</code> period.
<code>root_api_attack</code>	Configure as <code>true</code> if you want AI engine to detect attacks on the root API. Set it to <code>false</code> if you do not wish the AI engine to detect attacks on the root API. The default value is <code>false</code> .
<code>session_inactivity_duration</code>	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.  <b>Note:</b> 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({
  "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"
});
```

### 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 <code>true</code> , SSL communication is enabled between ASE and ABS, and for external systems making rest API calls to ABS. See <a href="#">Configure SSL</a> on page 172 on 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.

#### 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
```

You need to restart all the ABS node for your changes to take effect.

## 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",
  "continuous_learning": true,
  "url_limit": "100",
  "response_size": "100",
  "job_frequency" : "10",
  "window_length" : "24",
  "enable_ssl": true,
  "api_discovery": false,
  "discovery_initial_training" : "24",
  "discovery_subpath": "1",
  "discovery_update_interval": "1"
});
```

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.

**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 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
You need to restart all ABS nodes for your changes to take effect.
```

## 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>/v3/abs/attack/threshold?api=<api\\_name>](https://<ip_address>:<port>/v3/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 the [Attack Management Tool](#) 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 Tnvalue)

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  $T_n$  can never be more than the value of  $T_x$ .

### Changing Threshold Value $T_x$ only

Change the  $T_x$  value to adjust the gap between the normal and extreme threshold setting for an attack type on a specific API. The value of  $T_x$  defines the gap which ranges from a minimum of 1 to the maximum value defined in [Threshold range for  \$T\_n\$  and  \$T\_x\$](#) . When  $T_x$  is moved, the system calculated gap calculated at the end of the training period is no longer used. For the attack types where  $T_x$  is not applicable to the variable, “  $na$  ” is displayed in the threshold API.

**Note:** If the value of only  $T_n$  is moved without modifying  $T_x$ , then the new gap between the value of  $T_n$  and  $T_x$  is used until the value of  $T_x$  is changed again.

### Change threshold value $T_n$ and $T_x$ together

Both  $T_n$  and  $T_x$  can be changed for an attack type on a specific API. When  $T_n$  and  $T_x$  are moved simultaneously, the newly defined value of  $T_n$  and gap for  $T_x$  are changed. The ranges of  $T_n$  and  $T_x$  values are detailed in [Threshold range for  \$T\_n\$  and  \$T\_x\$](#) .

### How to configure threshold value

To manually set a threshold, use the PUT method with the following ABS attack API:

[https://<ip\\_address>:<port>/v3/abs/attack/threshold?api=<api\\_name>](https://<ip_address>:<port>/v3/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

In rare cases, an attack type may need to be completely disabled. 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 326 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.

## API discovery

API discovery consists of discovering new APIs and then automatically adding the APIs to using the Automatic API Definition (AAD) tool. The AAD tool is *installed* and configured separately.

The API Behavioral Security (ABS) AI Engine works in tandem with API Security Enforcer (ASE) to automatically discover APIs in your ecosystem. The discovery process works as follows:

1. ASE is configured with an API JSON file with `url` as `"/"` and `hostname` as `"*"`. ASE captures the API traffic metadata in log files and forwards API traffic to backend servers. It periodically sends the log files to ABS.
2. ABS processes the ASE log files and looks for new APIs. During the discovery period, ABS monitors the traffic on the API JSON (global API). At the end of the discovery period, the discovered APIs are reported. T0 to T24 in the diagram represents the discovery period.
3. At the end of the initial discovery period, ABS does the following:
  - If the API definition was learned, then ABS marks the API as *discovered*. Go to step 4.
  - If the API definition is incomplete, then ABS repeats the discovery process (Step 2) for a `discovery_update_interval` (default is 1-hour).
4. For each discovered API, the Automated API Definition (AAD) tool converts the ABS API definition report to ASE API JSON definition file format. AAD then adds the API JSON file to ASE.
5. When traffic is received from the new API, ABS AI engine begins a machine learning training process for an interval defined by `attacks_initial_training` to determine normal behavior. The diagram assumes this occurs immediately and is represented by T24 to T48 in the diagram.
6. After the training period completes, ABS AI engine can begin detecting attacks on the discovered APIs.

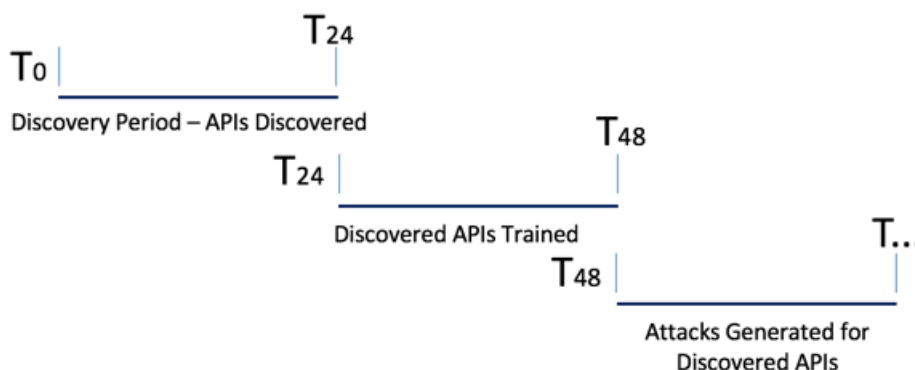
### API discovery variables

Variable	Description
<code>api_discovery</code>	Set this variable to <code>true</code> to switch on API discovery. To switch off API discovery, set it to <code>false</code> .
<code>discovery_initial_period</code>	The initial time in hours during which APIs are discovered in your API ecosystem. The API traffic must be continued for some time after the <code>discovery_initial_period</code> for APIs to be discovered.
<code>discovery_update_interval</code>	The time interval in hours at which any new discovered APIs are reported.
<code>discovery_subpath</code>	The number of subpaths that is discovered in an API. The minimum value is 1 and maximum value is 3. See <a href="#">Discovery Subpaths</a> on page 205 for more information.

Variable	Description
<i>url_limit</i>	This variables defines the number of URLs that are reported in a discovered API.

```
db.global_config.insert({
  "attacks_initial_training": "24",
  "attacks_update_interval": "24",
  "continuous_learning": true,
  "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",
  "discovery_update_interval": "1"
});
```

The following illustration shows the time line from the start of API discovery to the time when attack detection starts.



## Reporting the discovered APIs

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.

## 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 3. 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" : "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"
});
```

**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.

Update script help is available by logging into the MongoDB primary machine and running the following command:

```
/opt/pingidentity/mongo/update.sh help
```

**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=24
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.

## Root API in ASE and attack detection

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` 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
}

```

## 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:** </v3/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": "192.168.11.162",
      "basePath": "/pubatmapp",
      "created": "Wed Oct 25 20:31:46:082 2017",
      "updated": "Wed Oct 25 20:51:48:161 2017"
    },
    {
      "api_name": "api_1",
      "host": "192.168.11.162",
      "basePath": "/atmapp",
      "created": "Wed Oct 25 20:31:46:084 2017",
      "updated": "Wed Oct 25 20:51:48:158 2017"
    },
    {
      "api_name": "api_2",
      "host": "192.168.11.162",
      "basePath": "/app/ws",
      "created": "Wed Oct 25 20:31:46:086 2017",
      "updated": "Wed Oct 25 20:31:46:086 2017"
    }
  ]
}

```

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:** [/v3/abs/discovery?api\\_name=api\\_1](/v3/abs/discovery?api_name=api_1)

Here 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_1"
  },
  "host": "192.168.11.162",
  "basePath": "/atmapp",
  "schemes": [
    "http/1.1"
  ],
}

```

```

"consumes": [
  "application/json",
  "multipart/form-data"
],
"produces": [
  "text/html",
  "application/json",
  "text/plain"
],
"server_ssl": false,
"backendHosts": [
  "x.foo.backend1.com:3000",
  "x.foo.backend2.com:3000",
],
"backendServers": [
  "192.168.11.164:3001",
  "192.168.11.164:3002",
  "192.168.11.164:3003",
  "192.168.11.164:3004"
],
"paths": {
  "paths": {
    "/atmapp/zipcode": {
      "post": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      },
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      },
      "delete": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      },
      "put": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      }
    }
  }
}

```





**i Important:** 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 AAD should be configured for manual, not automatic injection of APIs into PingIntelligence.

## Manage discover 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 24-hours, then discovered APIs are reported 24-hours from the time when discovery was initiated.

### 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.

**i Note:** You need to stop and start all the ABS nodes for the updated values to take effect after the values are updated by executing the script.

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=24
url_limit=100
response_size=100
window_length=24
discovery_subpath=3
percentage_diskusage_limit=80
```

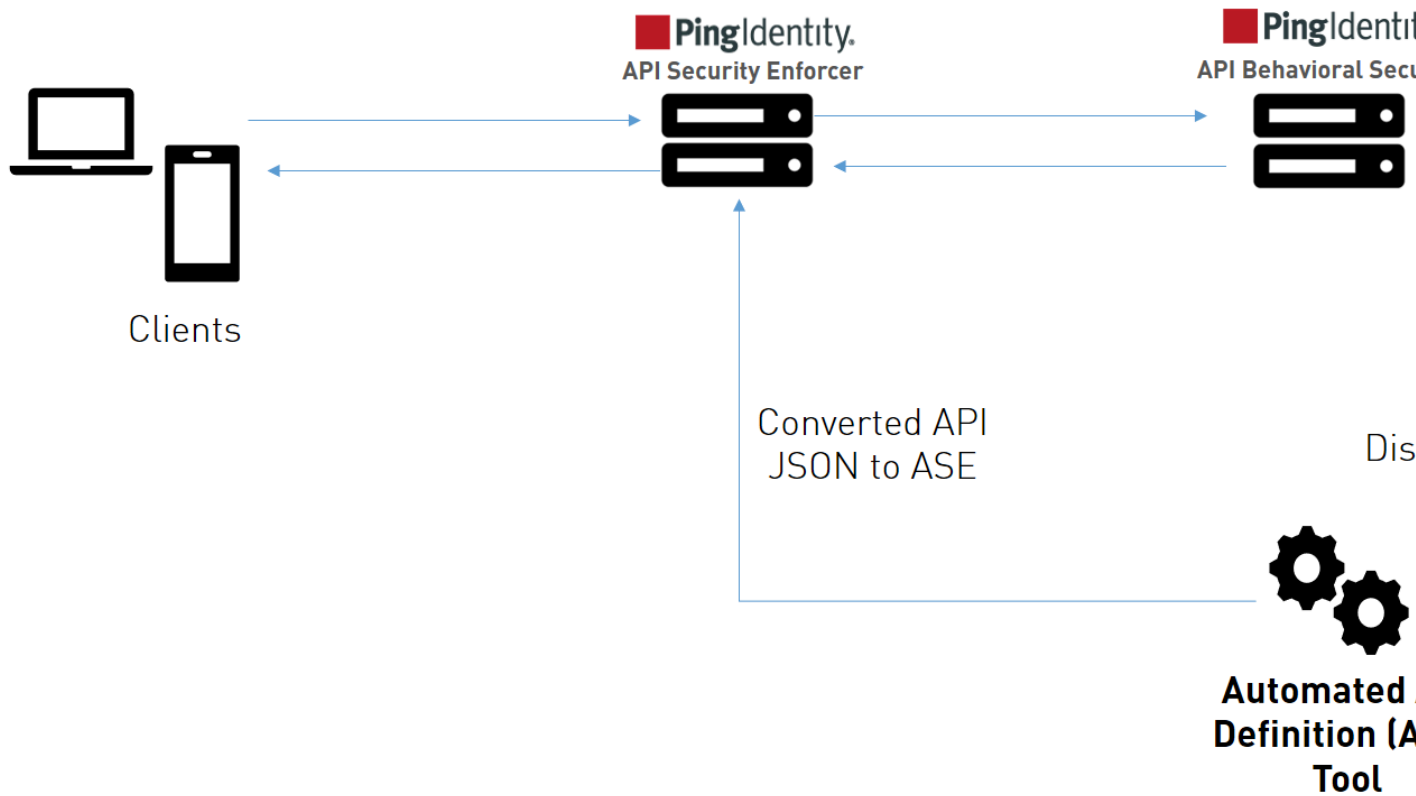
You need to restart all the ABS node for your changes to take effect.

## Automated API Definition (AAD)

Automated API Definition (AAD) is a Java-based tool that adds ABS discovered APIs to ASE. AAD also fetches API definitions from two supported API gateways, PingAccess and Axway. AAD runs independently of ABS but can run on the same machine as ABS. AAD works as follows:

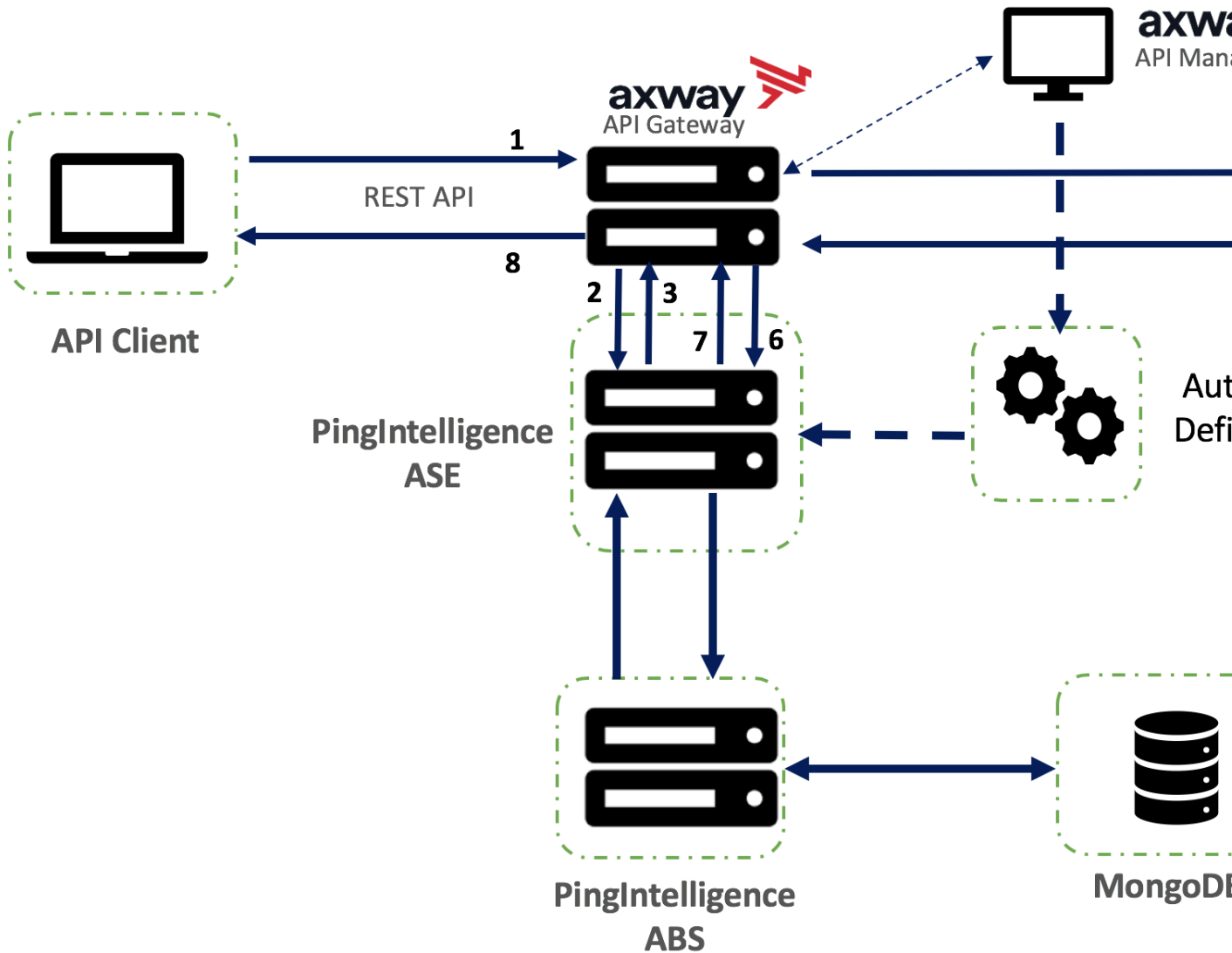
- AAD polls ABS at regular intervals for any newly discovered APIs.
- If new APIs are discovered, AAD converts the new API definitions to an API JSON file
- AAD then adds the API JSON file to ASE.

The following illustration summarizes the API discovery and feedback loop.



**Note:** Do not connect AAD to ASE if you do not want to send APIs discovered by ABS to ASE.

The following illustration summarizes the AAD fetching API definitions from Axway and adding them to ASE. It works in a similar way for PingAccess:



Automated API Definition Tool converts ABS discovered parameters to API JSON file format:

ABS Discovered API Parameters	AAD Converted API JSON Parameter
API ID: api_0	<hostname>_<url>
host	hostname
basePath	URL
schemes	protocol
consumes	NA
method	NA
backendServers	NA
server_ssl	server_ssl
backendHosts	servers

**Note:** In the converted API JSON, `api_memory_size` is set to 64mb, `health_check` as false and `enable_blockingsis` set to false by default. You can edit these values in ASE.

## Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD's logs
- `util` - Contains the `check_ports.sh`. Run on the machine with the AAD tool to check ASE and ABS default ports.

## Obfuscate keys and passwords

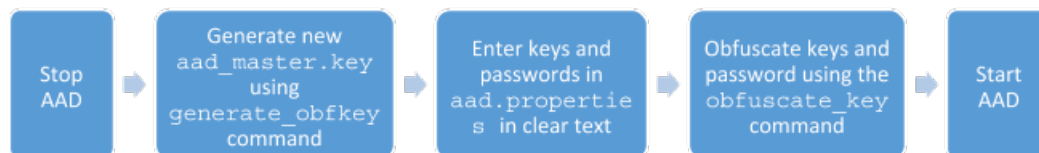
Using the AAD command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. The following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `abs.access_key`
- `abs.secret_key`
- `gateway.management.password`
- `pingaccess.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

**Note:** During the process of obfuscation of keys and password, AAD should not be running.

The following diagram summarizes the obfuscation process:



### Generate aad\_master.key

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI. The default password for admin user is `admin`.

```
/opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
```

```
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` should be moved to a secure location outside of AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
  config/aad.properties: ase.access_key, ase.secret_key,
  abs.access_key, abs.secret_key, gateway.management.password and
  pingaccess.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start [AAD](#) after passwords are obfuscated.

## AAD deployment options

Install and configure AAD to automatically capture API definitions. AAD discovers these API definitions, converts them into PingIntelligence API JSON files, and adds the definitions to ASE.

AAD supports two operating modes:

- Dev – the discovered API definitions are mirrored in the PingIntelligence environment. If an API is removed from the discovered APIs, it is deleted from PingIntelligence.
- Prod - the discovered API definitions are mirrored in the PingIntelligence environment but APIs are not deleted from PingIntelligence.

Use cases

Following are the use cases to explain the operating mode usage:

- **Development environment with partial synchronization**

When you set the `aad.dev` to `dev` (development environment) and `aad.env.dev.fullsync` to `false`, then AAD synchronizes the source APIs to ASE without deleting other APIs present in ASE.

**Example 1**

Let us assume that your development environment has an API called `api_at_source` and ASE has an API called `api_at_ase` *having same URL and hostname*. There are other APIs in ASE, let us say, `api_1`, `api_2` and so on. These APIs (`api_1`, `api_2`) are present in the source.

AAD in such a case would remove API `api_at_ase` in ASE and replace it with API `api_at_source` and its metadata. The other APIs in ASE are not deleted by AAD.

**Example 2**

Let us assume that your development environment and ASE has an API with same name, `api_x`. In this case AAD synchronizes the metadata of the API at source with that in the ASE. In this case also, the other APIs in ASE are not deleted

- **Development environment with complete synchronization**

When you set the value of `aad.dev` to `dev` (development environment) and that of `aad.env.dev.fullsync` to `true`, then AAD synchronizes the source APIs to ASE. AAD also deletes any API from ASE which is not present at the source.

**Note:** Note: AAD does not delete the root (`*/`) API from ASE.

**Example**

Let us assume that your development environment has APIs `api_1` and `api_2` while ASE has APIs `api_1`, `api_2`, and `api_3`. In this case, AAD synchronizes the metadata of AP `api_1` and `api_2` and deletes the API `api_3` from ASE.

- **Production environment**

When you set the value of `aad.dev` to `prod`, AAD works in the production environment mode. In this mode, AAD does not synchronize the changes made in APIs at the source. Use this mode when you want to add APIs to ASE by discovering using ABS or fetch APIs from API gateway. APIs once added to ASE by AAD are not updated in this mode.

Property	Description
<code>aad.env</code>	<p>Set to <code>dev</code> if you are fetching API definitions in an API development environment.</p> <p>Set the key to <code>prod</code> if you are fetching API definitions in a production environment</p>
<code>aad.env.dev.fullsync</code>	<p>Set it to <code>true</code> if you want to mirror source APIs in ASE. AAD deletes any APIs from PingIntelligence that are removed from the Source API definitions.</p> <p>Set it to <code>false</code> if you want to synchronize the source APIs in ASE without deleting other APIs present in ASE.</p>

**Note:** `aad.env.dev.fullsync` key is only valid in a development environment (`aad.env=dev`)

<code>aad.mode</code>	<p>Set the value to <code>discovery</code> when ASE is deployed in inline mode.</p> <p>Set the value to <code>gateway</code> when ASE is deployed in sideband mode with Axway gateway. For more information on AAD in <code>gateway</code> mode, see the <a href="#">PingIntelligence for APIs - Axway Integration</a>.</p> <p>Set the value to <code>pingaccess</code> when ASE is deployed in <code>sideband</code> mode with PingAccess. For more information on AAD in <code>pingaccess</code> mode, see the <a href="#">PingIntelligence for APIs – PingAccess Integration</a> guide.</p> <p>For more information on ASE modes, see the <a href="#">ASE Admin Guide</a>.</p>
-----------------------	---

## Configure AAD

Connecting AAD with ABS and ASE requires the following:

- ASE node hostname or IPv4 address and credentials
- ABS node hostname or IPv4 address and keys

These values are configured in the `aad.properties` file available in the `config` directory.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	<p>Set to <code>dev</code> if you are fetching API definitions in an API development environment.</p> <p>Set the key to <code>prod</code> if you are fetching API definitions in a production environment</p>
<code>aad.env.dev.fullsync</code>	<p>Set it to <code>true</code> if you want to mirror source APIs in ASE. AAD deletes any APIs from PingIntelligence that are removed from the Source API definitions.</p> <p>Set it to <code>false</code> if you want to synchronize the source APIs in ASE without deleting other APIs present in ASE.</p>

**Note:** `aad.env.dev.fullsync` key is only valid in a development environment (`aad.env=dev`)



aad.mode	<p>Set the value to <code>discovery</code> when ASE is deployed in inline mode.</p> <p>Set the value to <code>gateway</code> when ASE is deployed in sideband mode with Axway gateway. For more information on AAD in <code>gateway</code> mode, see the <a href="#">PingIntelligence for APIs - Axway Integration</a>.</p> <p>Set the value to <code>pingaccess</code> when ASE is deployed in <code>sideband</code> mode with PingAccess. For more information on AAD in <code>pingaccess</code> mode, see the <a href="#">PingIntelligence for APIs – PingAccess Integration</a> guide.</p> <p>For more information on ASE modes, see the <a href="#">ASE Admin Guide</a>.</p>
abs.host	<p>Hostname or IPv4 IP address of the ABS host machine.</p>
abs.port	<p>Port number of the ABS service.</p>
abs.access_key	<p>The access key of ABS. The default value is <code>abs_ak</code></p>
abs.secret_key	<p>The secret key of ABS. The default value is <code>abs_sk</code></p>
ase.secret_key	<p>The password of ASE. The default value is <code>admin</code></p>
ase.host	<p>Hostname or IPv4 IP address of the ASE host machine.</p>
ase.port	<p>Port number of the ASE service.</p>
abs.ssl	<p>Set to true for ABS-AAD communication to use SSL.</p>
ase.access_key	<p>The username of ASE. Default value is <code>admin</code></p>
abs.query.interval	<p>The polling interval in minutes to poll for any newly discovered APIs.</p> <p>The default value is 10 minutes.</p>
aad.log.level	<p>The log level of AAD log files. The default value is <code>INFO</code>. Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code></p>
gateway.management.url	<p>URL of the API Gateway.</p> <p>Only valid when <code>aad.mode</code> is <code>gateway</code>.</p>
gateway.management.username	<p>Username to connect to the API Gateway.</p> <p>Only valid when <code>aad.mode</code> is <code>gateway</code>.</p>
gateway.management.password	<p>Password to connect to the API Gateway.</p> <p>Only valid when <code>aad.mode</code> is <code>gateway</code>.</p>
pingaccess.management.url	<p>URL of the PingAccess.</p> <p>Only valid when <code>aad.mode</code> is <code>pingaccess</code>.</p>

<code>pingaccess.management.username</code>	Username to connect to the PingAccess. Only valid when <code>aad.mode</code> is <code>pingaccess</code> .
<code>pingaccess.management.password</code>	Password to connect to the PingAccess. Only valid when <code>aad.mode</code> is <code>pingaccess</code> .

Following is a sample `aad.properties` file:

```
# Automated API Discovery (AAD)
# Automated API Discovery (AAD)

# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod

# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
# or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E
### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsjTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSmv5mZllw3WpAsj1oPF3d5Et170Y=
# ABS secret key
```

```
abs.secret_key=OBF:AES:RsjTC/tx/sp
+7XXtr8+1rnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=
### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74LahxfEgp:dLfcNugFUCcfsglnBXQzflTvAWiPit8ulseHxi+Z0tk=
### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+lpEqcKQnFG/UN3Ezfz0DMA/
kmI=:Az82rlUFftMGpmxF7unelJZUucX1911O2QgKvHD36vU=
```

A sample API JSON for auto-discovery is shown below. The important fields to fill are:

- url - /
- hostname - \*
- server\_ssl – true or false based on whether your backend server is SSL enabled or not.
- servers – If an API gateway is behind ASE, then provide the hostname or the IP address of the API gateway. If the APIs are hosted as a service, then provide the hostname or the IP address of the server that hosts these servers.

```
{
  "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": "",
    "login_url": "",
    "enable_blocking": false,
    "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",
"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": [
  ]
}
}
}

```

## Start and stop AAD

### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

### Start AAD

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```

bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details

```

### Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```

bin/stop.sh
Ping Identity Inc.
AAD is stopped.

```

## Start AAD in offline mode

AAD can run in offline mode where fetched definitions are stored for review and not automatically added to PingIntelligence. In this mode AAD periodically fetches API definitions from ABS AI Engine (in discovery mode), converts the definitions to PingIntelligence API JSON files, and stores the definitions in a directory to allow for a manual review.

To start AAD in offline mode, run the start command with the `offline` option

```
./bin/start.sh --offline ./aad_offline
This will delete any existing files and directories in /home/sdappin/
pingidentity/aad/aad_offline
Do you want to continue (yes/no): yes
# Automated API Discovery
# starting AAD 4.0
2019-05-21 02:45:18 INFO - starting api conversion
2019-05-21 02:45:18 INFO - starting pre-conversion stage
2019-05-21 02:45:18 INFO - pre-conversion stage finished
2019-05-21 02:45:18 INFO - getting discovered apis from abs
2019-05-21 02:45:18 INFO - getting api discovery summary from abs
2019-05-21 02:45:18 INFO - found discovered api with api (id =api_0,
  host=172.16.40.165, url=/app), requesting api details to file ./
aad_offline/abs/api_0/abs_discovered_api.json
2019-05-21 02:45:18 ERROR - invalid backend host not available:0 for api
  id=api_0 from file ./aad_offline/abs/api_0/abs_discovered_api.json
2019-05-21 02:45:18 INFO - converting abs discovered apis to ase api format
2019-05-21 02:45:18 INFO - wrote api json body for api 172_16_40_165-
  app to file /opt/pingidentity/aad/./aad_offline/ase/172_16_40_165-
  app.json
```

**Note:** AAD creates an `ase` directory inside the specified destination. AAD converts the API definitions and stores the PingIntelligence API JSON files in the `<aad_offline>/ase` directory.

**CAUTION:** When AAD is run in offline mode, it clears the contents of the provided directory and saves the new API definitions.

## AAD Conversion Status

Check the status of API conversion by running the `status` command. To run the `status` command, navigate to `/opt/pingidentity/aad/bin` directory.

```
bin/status.sh
```

The status command has different output based on API conversion state:

- **Conversion stage:** Tool is converting the API. The status message is:

```
status : converting
details : APIs in conversion stage
```

- **Conversion successful:** APIs have been successfully converted. The status message is:

```
status : API conversion successful
details :
1) api_0 : successful, API converted to myhost_pubatmapp api, pushed to
  ASE
2) api_1 : successful, API converted to myhost_atmapp api, pushed to ASE
3) api_2 : successful, API converted to wshost_app api, pushed to ASE
```

- **Conversion unsuccessful:** APIs could not be converted. The status message is:

```
status : API conversion could not succeed
details : please see /opt/pingidentity/aad/logs/act.log for more details
```

- **Conversion partially successful:** Some APIs were converted successfully. The status message is:

```
status : API conversion partially successful
details : please see /opt/pingidentity/ aad /logs/act.log for more details
1) api_0 : successful, converted to myhost_pubatmapp api, pushed to ASE
```

```
2) api_1 : unsuccessful, conversion unsuccessful, incompatible api from  
ABS  
3) api_2 : unsuccessful, converted to wshost_app api, not able to push to  
ASE
```

- **AAD not running:** AAD is not running. Check the AAD log files for a possible reason.

The following table lists the input to the Automated API Definition Tool and the output API JSON.

## Input from ABS

```
{
  "company": "ping identity",
  "name":
    "api_discovery_details",
  "description": "This report
  contains
  details of discovered
  APIs",
  "info": {
    "title": "api_0"
  },
  "host": "myhost",
  "basePath": "/pubatmapp",
  "cookie": "JSESSIONID",
  "apikey_qs": "",
  "apikey_header": ""
  "oauth2_access_token": "false"
  "schemes": [
    "http/1.1"
  ],
  "consumes": [
    "application/json",
    "multipart/form-data",
    "text/plain"
  ],
  "produces": [
    "text/html",
    "application/json"
  ],
  "server_ssl": false,
  "backendHosts": [
    "x.foo.backend1.com:3000",
    "x.foo.backend2.com:3000"
  ],
  "backendServers": [
    "192.168.11.168:3000",
    "192.168.11.169:3000"
  ],
  "paths": {
    "/pubatmapp/update": {
      "put": {
        "produces": [
          "text/html"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      }
    },
    "/pubatmapp/login": {
      "post": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      }
    },
    "/pubatmapp/zipcode": {
      "post": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "ok"
          }
        }
      }
    }
  }
}
```

## Output to ASE

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/pubatmapp",
    "hostname": "myhost",
    "oauth2_access_token":
      false,
    "apikey_qs": "",
    "apikey_header": "",
    "enable_blocking": false,
    "cookie": "",
    "cookie_idle_timeout": "",
    "logout_api_enabled": false,
    "cookie_persistence_enabled":
      false,
    "login_url": "",
    "api_mapping": {
      "internal_url": ""
    },
    "api_pattern_enforcement": {
      "protocol_allowed": "",
      "http_redirect": {
        "response_code": "",
        "response_def": "",
        "https_url": ""
      },
      "methods_allowed": [],
      "content_type_allowed": "",
      "error_code": "",
      "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": [
      {
        "host":
          "x.foo.backend1.com",
        "port": 3000,
        "server_spike_threshold":
          "0/second",
        "server_connection_quota": 0
      },
      {
        "host":
          "x.foo.backend2.com",
        "port": 3000,
        "server_spike_threshold":
          "0/second",
        "server_connection_quota": 0
      }
    ],
    "decoy_config": {
      "decoy_enabled": false,
      "response_code": 200,
      "response_def": "",
      "response_message": ""
    }
  }
}
```

## Purge AAD log files

A `purge.sh` script either archives or purges processed access log files which are stored in the `/opt/pingidentity/aad/logs` directory.

**Note:** When the `purge` script is run, the log files are permanently deleted from the `/opt/pingidentity/aad/logs` directory. Always backup the files before deleting.

Located in the `/opt/pingidentity/aad/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/aad/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/aad/util/purge.sh -d 3
This will delete the data in /opt/pingidentity/aad/logs which is older than
3 days.
Are you sure (yes/no): yes
removing /opt/pingidentity/aad/logs/aad.log.2019-02-08 : last changed at Fri
Feb 8 23:51:09 EST 2019
removing /opt/pingidentity/aad/logs/aad.log.2019-02-09 : last changed at Sat
Feb 9 23:51:09 EST 2019
```

**Force delete:** You can force delete the AAD log files by using the `-f` option with `purge.sh` script. When you use the force purge option, the script does not check for confirmation to purge the log files. You can use the force purge option with the `-d` option to provide the number of days of logs you wish to keep.

**Example:** The following snippet shows the usage of force purge option with the `-d` option:

```
/opt/pingidentity/aad/util/purge.sh -d 2 -f
removing /opt/pingidentity/aad/logs/aad.log.2019-02-10 : last changed at Sun
Feb 10 00:11:55 EST 2019
removing /opt/pingidentity/aad/logs/aad.log.2019-02-11 : last changed at Mon
Feb 11 00:12:01 EST 2019
removing /opt/pingidentity/aad/logs/aad.log.2019-02-07 : last changed at Sat
Feb 9 00:12:27 EST 2019
Done.
```

In the above example, the script force purges the AAD 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/aad/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.

## 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>](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=2018-12-31T18:00&later\\_date=2018-10-25T13:30&api=shop&type=1](https://192.168.11.166:8080/v4/abs/attack?later_date=2018-12-31T18:00&later_date=2018-10-25T13:30&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	14
Data Poisoning	15
Data Exfiltration Attack Type 2	21
Content Scraping Type 2	28
Unauthorized Client Attack	29
Header inspection 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>](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=2018-12-31T18:00&later\\_date=2018-10-25T13:30&type=18](https://192.168.11.166:8080/v4/abs/attack?later_date=2018-12-31T18:00&later_date=2018-10-25T13:30&type=18)

The following table lists the attack types for individual APIs:

**Across API attacks**

<b>Attack Type</b>	<b>Type ID</b>
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)	22
ⓘ <b>Note:</b> Applicable only for Inline ASE deployment	
Excessive Client Connections (Token)	23
ⓘ <b>Note:</b> Applicable only for Inline ASE deployment	
Excessive Client Connections (IP)	24
ⓘ <b>Note:</b> Applicable only for Inline ASE deployment	
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
Probing Replay Attack - API Key	32
Extended Probing Replay Attack - API Key	33
Account Take over Type 1	34
Account Take over Type 2	35
Sequence Attack	36

**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 PingAccess or an API Gateway.

**Detected attacks based on username**

<b>Attack Type</b>	<b>Description</b>	<b>id</b>	<b>Single or Across APIs</b>
--------------------	--------------------	-----------	------------------------------

Account Take Over Type 1	Fraudulent user behavior which indicates that credentials are compromised.	34	Across APIs
Account Take Over Type 2	Fraudulent user behavior over an extended period which indicates that credentials are compromised	35	Across APIs
Sequence Attack	Highly abnormal sequence of API transactions	36	Across APIs

**i Important:** ABS can also report Sequence attacks based on a client's OAuth token activity. However, if a username is available, it is first reported against username.

## 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 - API Key	A stolen API Key is being used to attack an API service.	31	Across APIs
Probing Replay Attack - API Key	Probing or breach attempts on an API service using API Keys. This is also called fuzzing.	32	Across APIs
Extended Probing Replay Attack - API Key	Probing or breach attempts on an API service using API Keys over an extended period of time. This is also called fuzzing.	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	Session management service receiving an abnormal number of cookies from a client.	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
Extreme Client Activity Attack	Extreme client request activity on an API service.	10	Single API
Extreme App Activity	Extreme App Activity may indicate an injection or other CPU intensive attack.	11	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 or breach attempts on an API service – also called fuzzing.	18	Across APIs
Data Exfiltration Attack Type 2	Data is being extracted via a REST API service.	21	Single API
Excessive Client Connections	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	28	Single API

Header Inspection	Probing an API using malicious headers	37	Single API
-------------------	--	----	------------

## Attacks based on token activity

ABS AI Engine detects attack based on client activity using an OAuth Token. The following table lists the detected 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
Extreme Client Activity Attack	Extreme client request activity on an API service.	10	Single API
Extreme App Activity	Extreme App Activity may indicate an injection or other CPU intensive attack.	11	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 or breach attempts on an API service – also called fuzzing.	19	Across APIs
Data Exfiltration Attack Type 2	Data is being extracted via a REST API service.	21	Single API
Excessive Client Connections	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	28	Single API
Sequence Attack	Abnormal sequence of transactions	36	Across APIs
Header Inspection	Probing an API using malicious headers	37	Single API

**i 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 IP address as the main 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
Extreme Client Activity Attack	Extreme client request activity on an API service.	10	Single API
Extreme App Activity	Extreme App Activity may indicate an injection or other CPU intensive attack.	11	Single API
API DoS Attack	Inbound client request limits exceeded on an API service.*	12	Single API
API DDoS Attack Type 2	A DDoS or distributed attack is overloading an 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 or breach attempts on an API service – also called fuzzing.	20	Across APIs
Data Exfiltration Attack Type 2	Data is being extracted via a REST API service.	21	Single API
Excessive Client Connections	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	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.	30	Across APIs
Header Inspection	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

## 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.

### Automated 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, 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 block future access for clients using 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
```

## Whitelist and blacklist management

ASE maintains two types of lists:

- **Whitelist** – List of “safe” IP addresses, cookies, OAuth2 Tokens or API keys that will not be blocked by ASE. The list is manually generated by CLI commands.
- **Blacklist** – List of “bad” IP addresses, cookies, OAuth2 Tokens or API keys 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)
  - List of “bad” clients manually generated by CLI

ABS manages a list which includes ABS and ASE clients suspected of attacks. However, ABS does not receive manually generated lists (for example, white list, imported black lists).

#### Manage Whitelist

Valid ASE operations for OAuth2 Tokens, Cookies, IP addresses 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
```

- **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
```

- **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 Blacklist

Valid ASE operations for IP addresses, Cookies, OAuth2 Tokens 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 ad233edqsdld23redwefew
cookie JSESSIONID ad233edqsdld23redwefew added to blacklist
```

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token
ad233edqsdld23redwefew
token ad233edqsdld23redwefew 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
```

- **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.

## TTL for client identifiers

The ABS AI Engine blacklist supports configuring the length of time that a client identifier type (i.e. username, OAuth token, API Key, cookie, IP address) remains on the blacklist. Each client identifier type can be configured with a different value in minutes. Since the default value is zero, 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 entity type. 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.

**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",
  "across_api_prediction_mode": false,
  "api_discovery": {
    "status": false
  },
  "apis": [
    {
      "api_name": "app",
      "host_name": "*",
      "url": "/atm_app_oauth",
      "api_type": "decōy-incontext",
      "creation_date": "Thu May 02 09:51:10 UTC 2019",
      "servers": 0,
      "protocol": "http",
      "cookie": "",
      "token": true,
      "training_started_at": "Thu May 02 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 May 01 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 May 02 10:51:13 UTC 2019",
      "logs": {
        "start_time": "Thu May 02 09:51:14 UTC 2019",
        "end_time": "Thu May 02 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 May 02 10:59:40 UTC 2019"
}

```

To verify the new TTL values, rerun the ABS `admin` REST API with the GET method.

## ABS blacklist management and 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 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.

**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 client identifiers from blacklist

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. 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.

You can also use the [Attack Management Tool](#) to delete client identifiers from ASE and ABS. For more information, see [Unblock clients from ABS and ASE](#).

## 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 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 Management Tool

Attack Management Tool (AMT) simplifies the removal of client identifiers from the blacklist and supports tuning of AI model thresholds to reduce the chances of future false positives. Download the Attack Management Tool from the Ping's [download](#) site. You can install AMT on the same machine as AAD.

Install Attack Management Tool

**Prerequisite:** To install AMT, the machine should have OpenJDK 11 installed.

To install AMT, complete the following steps:

1. Download and save the AMT tool on your machine
2. Untar the file to install the tool:

```
# tar -zxf amt-4.0.tar.gz
```

Installing AMT creates the following directory structure:

Directory	Description
bin	Contains the <code>cli.sh</code> script to run the various AMT commands.
config	Contains <code>amt.properties</code> and <code>amt_master.key</code> .
lib	System directory. Do not edit.
logs	Contains AMT log files.

Configure AMT

Configure AMT by editing the `amt.properties` file in the `config` directory. The following table lists the `amt.properties` variables:

Variable	Description
<code>amt.log.level</code>	The log level of AMT log files. The default value is <code>INFO</code> . Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code>
<b>ASE Configuration</b>	
<code>ase.host</code> and <code>ase.port</code>	ASE hostname or IPv4 address and ASE port number.
<code>ase.access_key</code> and <code>ase.secret_key</code>	ASE access key (username) and secret key (password). The default value is <code>admin/admin</code> . These values are obfuscated in the default <code>amt.properties</code> file.
<b>ABS Configuration</b>	
<code>abs.host</code> and <code>abs.port</code>	ABS hostname or IPv4 and ABS port number.
<code>abs.access_key</code> and <code>abs.secret_key</code>	ABS access key (default is <code>abs_ak</code> ) and secret key (default is <code>abs_sk</code> ) These values are obfuscated in the default <code>amt.properties</code> file.

Following is a sample `amt.properties` file:

```
# Attack Management Tool (AMT)
# 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.

# Log level
amt.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:F3XGSgx21I5TMRy39+fMfvf
+Jxky:uulOfTXINmNMLzMakpgBHb4ZmVFhUaFobUBsM+n8jxQ=
# ASE REST API secret key
ase.secret_key=OBF:AES:F3XGSgx21I5TMRy39+fMfvf
+Jxky:uulOfTXINmNMLzMakpgBHb4ZmVFhUaFobUBsM+n8jxQ=
### ABS config.
# ABS Host ( hostname or IPv4 address )
abs.host=172.16.40.208
# ABS management port
abs.port=8080
# ABS access key
abs.access_key=OBF:AES:F3PYfANEC4K+l0vdAaWmdiNu5heOrQ==:fx0VMDIZFXHCjaoHDio/
QFkFqgMbY0VMX4FLDxlEIH4=
# ABS secret key
abs.secret_key=OBF:AES:F3PYfBFehdA3Mk0poxzqZgimpdg9RA==:8EE
+REe9eNFgjSycnW31V5gJgH8qmgBi2g5zDUNjcOE=

```

## Obfuscate keys

Obfuscate ASE and ABS keys. AMT ships with a default master key. You can generate your own master key. The master key is used to obfuscate the keys in `amt.properties` file.

**Generate master key:** Run the `generate_obfkey` command to generate a new master key.

```

./bin/cli.sh generate_obfkey
Please take a backup of /home/amt/config before proceeding.
Once you create a new obfuscation master key, you should obfuscate all
config keys.
Warning: Obfuscation key file already exists. This command will delete it
and create a new key in the same file

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

Generating obfuscation key
Obfuscation key created successfully.

```

**Obfuscate keys:** Run the `obfuscate_keys` command to obfuscate the keys in `amt.properties` file:

```

Please take a backup of /opt/pingidentity/amt/config/amt.properties before
proceeding. This will obfuscate the following properties:

```

```
ase.access_key
ase.secret_key
abs.access_key
abs.secret_key

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

Obfuscating properties in amt.properties
Obfuscation successful
```

## Unblock client

You can unblock clients by using AMT to remove the client identifiers from the blacklist. Run the `unblock_client` command to delete a specific client identifier. The interactive `unblock_client` command asks for different inputs like, the component (ASE or ABS) from where you want to delete the client identifier, the type of client identifier (IP, cookie, token, API key, or username) and its value. You can delete only one client identifier at a time. Following is a sample output of the `unblock_client` command:

```
./bin/cli.sh unblock_client
-> Choose the client identifier type.
1) ip
2) cookie
3) access_token
4) api_key
5) username
-> Enter [1-5]: 1
-> Enter ip value: 192.168.12.63
-> Deleting attacks from ABS
No entries were deleted. Attack doesn't exist in ABS attack list.
-> Deleting attacks from ASE
ip 192.168.12.63 deleted from blacklist
```

## Tune thresholds

Attack management tool can tune AI engine thresholds to reduce the chance of future false positives. The attack report provides the threshold values used to detect an attack, and AMT queries the values to automatically determine the necessary threshold adjustment. The `tune_threshold` command asks for the type and value of the client identifier identified executing an attack. The attack management tool identifies the attacks associated with the client identifier and tunes its threshold. Here is a sample run of the `tune_threshold` command:

```
./bin/cli.sh tune_threshold
-> Choose the client identifier type:
1) ip
2) cookie
3) access_token
4) api_key
5) username
-> Enter [1-5]: 1
-> Enter the client identifier value: 100.100.249.1
-> Processing Attack id: Content Scraping Type 2, and api: shop-books
-> Processing Attack id: Data Exfiltration Attack Type 1, and api: shop-
books
-> Setting attack threshold for API: shop-books, for attack id: 1, for
variable: A
{"message":"success: new attack threshold is updated.,"date":"Wed Jun 12
06:04:37 MDT 2019"}
-> Setting attack threshold for API: shop-books, for attack id: 28, for
variable: A
{"message":"success: new attack threshold is updated.,"date":"Wed Jun 12
06:04:38 MDT 2019"}
```

```
-> Setting attack threshold for API: shop-books, for attack id: 28, for
variable: B
{"message":"success: new attack threshold is updated.", "date":"Wed Jun 12
06:04:39 MDT 2019"}
```

For more information on thresholds, see [Tune thresholds for false positives](#).

## 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>](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](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>/v3/abs/attack?later\\_date=yyyy-mm-ddThh:mm&later\\_date=yyyy-mm-ddThh:mm&type=<type\\_id>](https://<ABS_IP:port>/v3/abs/attack?later_date=yyyy-mm-ddThh:mm&later_date=yyyy-mm-ddThh:mm&type=<type_id>).

**Example:** [https://192.168.11.166:8080/v3/abs/attack?later\\_date=2018-12-31T18:00&later\\_date=2018-10-25T13:30&type=0](https://192.168.11.166:8080/v3/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": "BDGC519055KGG4HR",
        "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)"
      }
    ]
  }
]
}
]

```

## API Deception

### 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](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/v3/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.

## 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](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.

**Note:** A Java sample client to view the result using the metrics and anomaly API is available on Ping Identity's download site.

Here is an excerpt 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"
]
}
]
},
```

## 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>](http://<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/v3/abs/bc?later\\_date=<>T<hh:mm>&earlier\\_date=<>T<hh:mm>&details=true](http://<ABS_IP>:port/v3/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_datapoisoning",
      "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"
]
}
]
}
]
}

```



```

    }
  }, {
    "ip": "127.0.0.1",
    "total_requests": 15,
    "devices": {
      "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/57.0.2987.110 Safari/537.36": 15
    },
    "methods": {
      "POST": 15
    },
    "urls": {
      "/atm_app_oauth": 15
    },
    "apis": {
      "atm_app_oauth": 15
    }
  }
]
}, {
  "username": "t5",
  "tokens": ["t5xsULXD", "t5WbMcCG", "t5qXGtcz"],
  "total_requests": 30,
  "ip_list": [{
    "ip": "127.0.0.28",
    "total_requests": 15,
    "devices": {
      "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/57.0.2987.110 Safari/537.36": 15
    },
    "methods": {
      "POST": 15
    },
    "urls": {
      "/atm_app_oauth": 15
    },
    "apis": {
      "atm_app_oauth": 15
    }
  }
], {
  "ip": "127.0.0.1",
  "total_requests": 15,
  "devices": {
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/57.0.2987.110 Safari/537.36": 15
  },
  "methods": {
    "POST": 15
  },
  "urls": {
    "/atm_app_oauth": 15
  },
  "apis": {
    "atm_app_oauth": 15
  }
}
]
}, {
  {
    "username": "t1",
    "tokens": ["t1TbzDxT", "t1lucvKoX", "t1RXrzNH"],
    "total_requests": 30,
    "ip_list": [{
      "ip": "127.0.0.28",
      "total_requests": 15,
      "devices": {

```

```

    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/57.0.2987.110 Safari/537.36": 15
  },
  "methods": {
    "POST": 15
  },
  "urls": {
    "/atm_app_oauth": 15
  },
  "apis": {
    "atm_app_oauth": 15
  }
}, {
  "ip": "127.0.0.1",
  "total_requests": 15,
  "devices": {
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/57.0.2987.110 Safari/537.36": 15
  },
  "methods": {
    "POST": 15
  },
  "urls": {
    "/atm_app_oauth": 15
  },
  "apis": {
    "atm_app_oauth": 15
  }
}
}
  {
    "ip": "127.0.0.1",
    "total_requests": 15,
    "devices": {
      "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
      Chrome/57.0.2987.110 Safari/537.36": 15
    },
    "methods": {
      "POST": 15
    },
    "urls": {
      "/atm_app_oauth": 15
    },
    "apis": {
      "atm_app_oauth": 15
    }
  }
}
}
}
}

```

## 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_typememory",
  "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&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

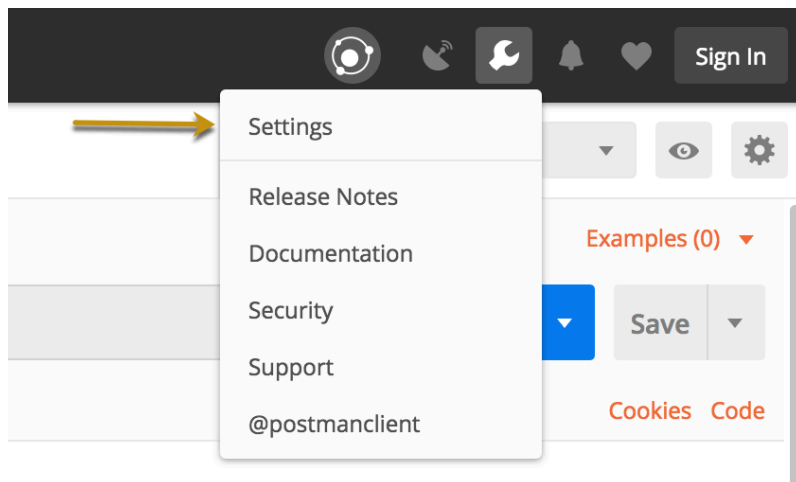
ABS ships with a self-signed certificate. If you want to use Postman with the self-signed certificate of ABS, then from Postman's settings, disable the certificate verification option. Complete the following steps to disable Postman from certificate verification:

1.

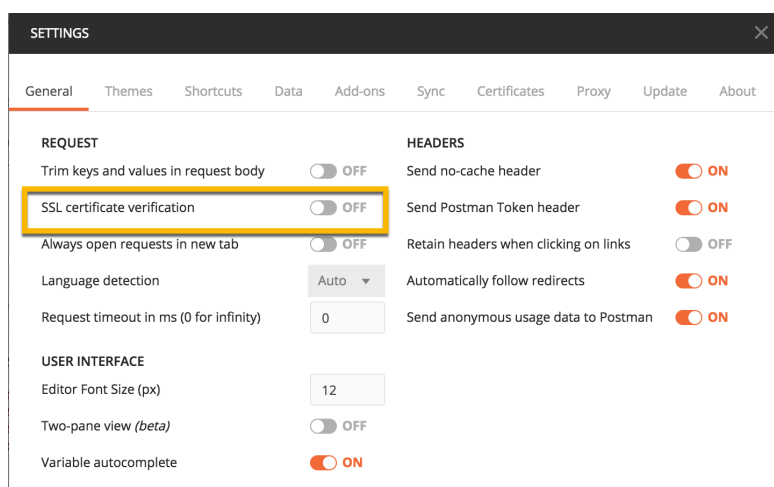


Click on the **spanner** on the top-right corner of Postman client. A drop-down window is displayed.

2. Select **Settings** from the drop-down window:



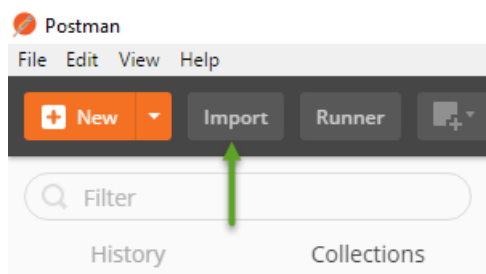
3. In the Settings window, switch-off certificate verification by clicking on the SSL certificate verification button:




## View ABS reports in Postman

To view the reports, complete the following steps:

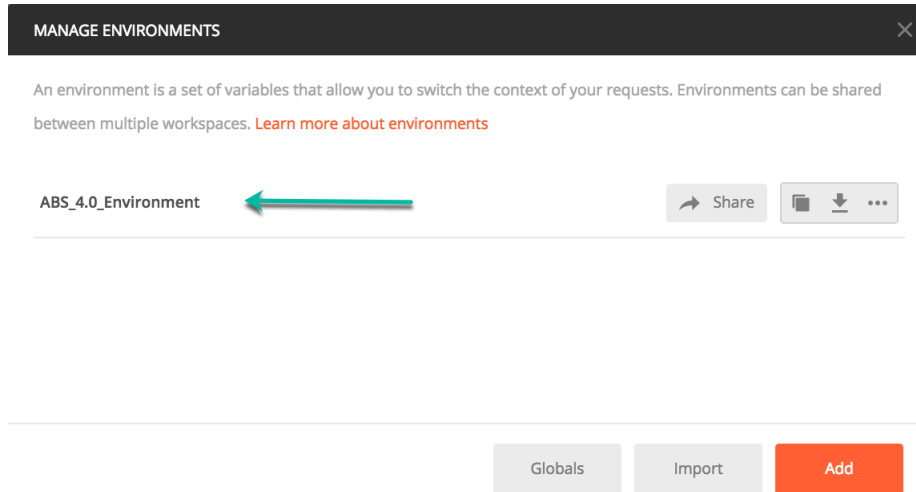
1. Download `ABS_4.0_Environment` and `ABS_4.0_Reports` JSON files from **API Reports Using Postman** folder on Ping Identity [Download](#) site. These configuration files will be used by Postman.
2. [Download](#) and install the Postman application 6.2.5 or higher.
3. In Postman, **import** the two 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\_4.0\_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`.

MANAGE ENVIRONMENTS
✕

Environment Name

ABS\_4.0\_Environment

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ		Persist All	Reset All
<input checked="" type="checkbox"/>	Server	192.168.11.166	192.168.11.166			
<input checked="" type="checkbox"/>	Port	8080	8080			
<input checked="" type="checkbox"/>	Access_Key_Header	x-abs-ak	x-abs-ak			
<input checked="" type="checkbox"/>	Secret_Key_Header	x-abs-sk	x-abs-sk			
<input checked="" type="checkbox"/>	Access_Key	abs_ak	abs_ak			
<input checked="" type="checkbox"/>	Secret_key	abs_sk	abs_sk			
<input checked="" type="checkbox"/>	API_Name	atmapp	atmapp			
<input checked="" type="checkbox"/>	Later_Date	2019-05-28T12:00	2019-05-28T12:00			
<input checked="" type="checkbox"/>	Earlier_Date	2019-05-22T12:00	2019-05-22T12:00			
<input checked="" type="checkbox"/>	System_URL	https://{{Server}}:{{P...	https://{{Server}}:{{Port}}/v4/abs			
<input checked="" type="checkbox"/>	System_Admin	{{System_URL}}/admin	{{System_URL}}/admin			
<input checked="" type="checkbox"/>	System_Metrics	{{System_URL}}/met...	{{System_URL}}/metrics?later_date={{Later_Date}}&earl...			
<input checked="" type="checkbox"/>	System_API_Key_Met...	{{System_URL}}/apik...	{{System_URL}}/apikeyes?later_date={{Later_Date}}&earl...			
<input checked="" type="checkbox"/>	System_OAuth_Toke...	{{System_URL}}/oaut...	{{System_URL}}/oauthtokens?later_date={{Later_Date}}...			
<input checked="" type="checkbox"/>	System_Anomalies	{{System_URL}}/ano...	{{System_URL}}/anomalies?later_date={{Later_Date}}&...			
<input checked="" type="checkbox"/>	System_Flow_Control	{{System_URL}}/flow...	{{System_URL}}/flowcontrol?later_date={{Later_Date}}&...			
<input checked="" type="checkbox"/>	System_Attack_Type	{{System_URL}}/atta...	{{System_URL}}/attack?later_date={{Later_Date}}&earli...			
<input checked="" type="checkbox"/>	System_Attack_Type...	{{System_URL}}/atta...	{{System_URL}}/attack/threshold?api={{API_Name}}			
<input checked="" type="checkbox"/>	System_Attack_Type...	{{System_URL}}/atta...	{{System_URL}}/attack/threshold			
<input checked="" type="checkbox"/>	System_Backend_Err...	{{System_URL}}/be?l...	{{System_URL}}/be?later_date={{Later_Date}}&earlier_d...			

i
Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#)
✕

Cancel

Update

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 and AAD CLI

ABS and AAD CLI provides the commands listed in the following table. The commands to obfuscate passwords, to generate the master and to update the admin password are the same for ABS and AAD.

### Basic commands

- [Start ABS](#)
- [Stop ABS](#)
- [Start AAD](#)

- [Stop AAD](#)
- [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
```

### Start AAD

#### Description

Starts AAD. Run the command from `/opt/pingidentity/abs/bin` directory

#### Syntax

```
./start.sh
```

### Stop AAD

**Description**

Stops AAD. Run the command from /opt/pingidentity/abs/bin directory

```
./stop.sh
```

## 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](#).

- [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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	x-abs-sk	<string>

### Sample Response

```
{
  "company": "ping identity",
```

```

"name": "api_admin",
"description": "This report contains status information on all APIs, ABS
clusters, and ASE logs",
"across_api_prediction_mode": true,
"api_discovery": {
  "status": false
},
"apis": [
  {
    "api_name": "rist_api",
    "host_name": "*",
    "url": "/rist",
    "api_type": "regular",
    "creation_date": "Mon Jun 10 06:56:52 UTC 2019",
    "servers": 2,
    "protocol": "http",
    "cookie": "",
    "token": false,
    "training_started_at": "Mon Jun 10 07:02:31 UTC 2019",
    "training_duration": "1 hour",
    "prediction_mode": true,
    "apikey_header": "",
    "apikey_qs": "Y-API-KEY",
    "is_active": true
  },
  {
    "api_name": "atm_app",
    "host_name": "*",
    "url": "/atm_app",
    "api_type": "regular",
    "creation_date": "Mon Jun 10 06:56:52 UTC 2019",
    "servers": 1,
    "protocol": "http",
    "cookie": "",
    "token": false,
    "training_started_at": "Mon Jun 10 07:02:31 UTC 2019",
    "training_duration": "1 hour",
    "prediction_mode": true,
    "apikey_header": "X-API-KEY",
    "apikey_qs": "",
    "is_active": true
  },
  {
    "api_name": "rest_api",
    "host_name": "*",
    "url": "/rest",
    "api_type": "regular",
    "creation_date": "Mon Jun 10 06:56:52 UTC 2019",
    "servers": 2,
    "protocol": "http",
    "cookie": "",
    "token": false,
    "training_started_at": "Mon Jun 10 07:02:31 UTC 2019",
    "training_duration": "1 hour",
    "prediction_mode": true,
    "apikey_header": "",
    "apikey_qs": "",
    "is_active": true
  },
  {
    "api_name": "root_api",
    "host_name": "*",
    "url": "/root",
    "api_type": "regular",

```

```

    "creation_date": "Mon Jun 10 06:56:52 UTC 2019",
    "servers": 2,
    "protocol": "http",
    "cookie": "",
    "token": false,
    "training_started_at": "Mon Jun 10 07:02:31 UTC 2019",
    "training_duration": "1 hour",
    "prediction_mode": true,
    "apikey_header": "",
    "apikey_qs": "Z-API-KEY",
    "is_active": true
  }
],
"abs_cluster": {
  "abs_nodes": [
    {
      "node_ip": "172.16.40.185",
      "os": "Red Hat Enterprise Linux Server - VMware, Inc.",
      "cpu": "16",
      "memory": "31G",
      "filesystem": "1%",
      "bootup_date": "Mon Jun 10 06:43:24 UTC 2019"
    },
    {
      "node_ip": "172.16.40.184",
      "os": "Red Hat Enterprise Linux Server - VMware, Inc.",
      "cpu": "16",
      "memory": "31G",
      "filesystem": "1%",
      "bootup_date": "Mon Jun 10 06:44:16 UTC 2019"
    }
  ],
  "mongodb_nodes": [
    {
      "node_ip": "172.16.40.181",
      "status": "up"
    },
    {
      "node_ip": "172.16.40.177",
      "status": "up"
    },
    {
      "node_ip": "172.16.40.178",
      "status": "up"
    }
  ]
},
"ase_logs": [
  {
    "ase_node": "2d03a149-3755-4224-adcf-a8c399a135c9",
    "last_connected": "Mon Jun 10 11:41:52 UTC 2019",
    "logs": {
      "start_time": "Mon Jun 10 06:57:01 UTC 2019",
      "end_time": "Mon Jun 10 11:41:52 UTC 2019",
      "gzip_size": "2.65MB"
    }
  }
],
"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
        }
    ]
}
}

```

## 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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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": "192.168.12.9",

```



```

"base_path": "/shop_bp",
"created": "Sun Jan 08 21:42:10:973 2018",
"updated": "Sun Jan 08 22:02:12:243 2018"
},
{
"api_name": "api_1",
"host": "192.168.12.13",
"base_path": "/bill_bp",
"created": "Sun Jan 08 21:42:10:974 2018",
"updated": "Sun Jan 08 22:22:22:393 2018"
},
{
"api_name": "api_2",
"host": "192.168.12.18",
"base_path": "/cart_bp",
"created": "Sun Jan 08 21:42:10:976 2018",
"updated": "Sun Jan 08 22:02:12:249 2018"
},
{
"api_name": "api_3",
"host": "192.168.12.20",
"base_path": "/login_bp",
"created": "Sun Jan 08 21:42:10:977 2018",
"updated": "Sun Jan 08 22:02:12:251 2018"
},
}
}

```

## 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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
  }
}
]
}
]
}
}

```

## GET Threshold REST API

**Description:** The GET 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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	x-abs-sk	<string>

## Sample Response for an API

```

{
  "company": "ping identity",
  "name": "api_threshold",
  "description": "This report contains threshold settings for the
  specified API",
  "api_name": "shop",
  "threshold": [
    {
      "type": "api_ddos_attack_type_1",
      "system": {
        "A": {
          "tn": "2",
          "tx": "7"
        }
      }
    },
    {
      "type": "api_dos_attack",
      "system": {

```

```

"A": {
  "tn": "5",
  "tx": "na"
}
},
{
  "type": "api_memory_attack_type_1",
  "system": {
    "A": {
      "tn": "10",
      "tx": "12"
    },
    "B": {
      "tn": "3",
      "tx": "5"
    },
    "C": {
      "tn": "2",
      "tx": "4"
    }
  }
},
{
  "type": "api_memory_attack_type_2",
  "system": {
    "A": {
      "tn": "9",
      "tx": "11"
    },
    "B": {
      "tn": "3",
      "tx": "5"
    },
    "C": {
      "tn": "2",
      "tx": "4"
    }
  }
},
{
  "type": "api_probing_replay_attack",
  "system": {
    "A": {
      "tn": "2",
      "tx": "4"
    }
  }
},
{
  "type": "data_exfiltration_attack",
  "system": {
    "A": {
      "tn": "22",
      "tx": "24"
    },
    "B": {
      "tn": "4",
      "tx": "6"
    },
    "C": {
      "tn": "-1",
      "tx": "-1"
    }
  }
}

```

```

}
},
{
  "type": "data_poisoning_attack",
  "system": {
    "A": {
      "tn": "9",
      "tx": "11"
    },
    "B": {
      "tn": "4",
      "tx": "6"
    },
    "C": {
      "tn": "2",
      "tx": "4"
    }
  }
},
{
  "type": "extreme_client_activity_attack",
  "system": {
    "A": {
      "tn": "5",
      "tx": "7"
    }
  }
},
{
  "type": "extreme_system_response_time",
  "system": {
    "A": {
      "tn": "2",
      "tx": "4"
    }
  }
},
{
  "type": "multi_client_login_attack",
  "system": {
    "A": {
      "tn": "34",
      "tx": "na"
    }
  }
},
{
  "type": "single_client_login_attack",
  "system": {
    "A": {
      "tn": "4",
      "tx": "6"
    },
    "B": {
      "tn": "4",
      "tx": "6"
    }
  }
},
{
  "type": "stolen_cookie_token_attack",
  "system": {
    "A": {
      "tn": "2",

```

```

    "tx": "na"
  },
  "B": {
    "tn": "5",
    "tx": "7"
  },
  "C": {
    "tn": "2",
    "tx": "na"
  }
}
]
}

```

### 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 REST API

**Description:** The PUT 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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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:

```
{
  "message": "success: new attack threshold is updated.",
  "date": "Wed Dec 05 14:26:41 IST 2018"
}
```

### 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:

```
{
  "message": "success: new attack threshold is updated.",
  "date": "Wed Dec 05 14:12:47 IST 2018"
}
```

## Metrics REST API

**Description** The 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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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",

```



```

{
  "api_key": "87FYNG7Q8KP1V030",
  "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
        }
      }
    ]
  }
}

```



```

"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,

```



## 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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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",
        "tONSwIjU",
        "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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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_exfiltration_VH",
"access_time": [
"Mon Jan 13 04:54:49:222 2018"
]
}
},
{
"cookie": "data_exfiltration_H",
"access_time": [
"Mon Jan 13 05:26:53:981 2018"
]
}
]
},
{
"anomaly_type": "Cookie DoS Attack",
"cookies": [
{
"cookie": "data_exfiltration_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_exfiltration__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"
]
}
}

```



```

    },
    {
      "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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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",

```





```

"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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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?dynamic_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
<b>Access Key</b>	x-abs-ak	<string>
<b>Secret Key</b>	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"
  ]
}
]
}

```

## Threshold range for Tn and Tx

---

Threshold range for Tn and Tx

The following table details the range of  $T_n$  and  $T_x$  for each attack type. When manually adjusting the threshold values, the values must fall within these range.

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)	Variable D (Range)	Variable E (Range)	Variable F (Range)
<b>REST API</b>							
Data Exfiltration	1	$T_n = [1-32]$ $T_x = [2-33]$	$T_n = [1-19]$ $T_x = [2-20]$	$T_n = [1-99]$ $T_x = [2-100]$	NA	NA	NA
Single Client Login	2	$T_n = [1-19]$ $T_x = [2-20]$	$T_n = [1-19]$ $T_x = [2-20]$	NA	NA	NA	NA
Multi Client Login	3	$T_n = [1-100]$ $T_x = \text{"na"}$	NA	NA	NA	NA	NA
Stolen Cookie / Access Token	4	$T_n = [2-10]$	$T_n = [1-19]$ , $T_x = [2-20]$	$T_n = [2-10]$	NA	NA	NA
API Memory Attack Type 1	5	$T_n = [1-32]$ $T_x = [2-33]$	$T_n = [1-19]$ $T_x = [2-20]$	$T_n = [1-99]$ $T_x = [2-100]$	NA	NA	NA
API Memory Attack Type 2	6	$T_n = [1-32]$ $T_x = [2-33]$	$T_n = [1-19]$ $T_x = [2-20]$	$T_n = [1-99]$ $T_x = [2-100]$	NA	NA	NA
Cookie DoS	7	$T_n = [1-99]$ $T_x = [2-100]$	$T_n = [1-19]$ $T_x = [2-20]$	NA	NA	NA	NA
API Probing Replay	8	$T_n = [1-99]$ $T_x = [2-100]$	NA	NA	NA	NA	NA
API DoS Attack Type 1	9	$T_n = [1-100]$ $T_x = \text{"[2-100]"}$	NA	NA	NA	NA	NA
Extreme Client Activity	10	$T_n = [1-19]$ $T_x = [2-20]$	NA	NA	NA	NA	NA
Extreme App Activity	11	$T_n = [1-19]$ $T_x = [2-20]$	NA	NA	NA	NA	NA
API DoS Attack	12	$T_n = [1-100]$ $T_x = \text{"na"}$	NA	NA	NA	NA	NA
API DDoS Attack Type 2	13	NA	NA	NA	NA	NA	NA
Data Deletion	14	$T_n = [1-19]$ $T_x = [2-20]$	$T_n = [1-99]$ $T_x = [2-100]$	NA	NA	NA	NA

Data Poisoning	15	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	Tn = [1-32] Tx = [2-33]	NA	NA	NA
Stolen Token Attack Type 2	16	Tn = [2-10] Tx = "na"	Tn = [1-100]	Tn = [1-100]	NA	NA	NA
Stolen Cookie Attack Type 2	17	Tn = [2-10] Tx = "na"	Tn = [1-100]	Tn = [1-100]	NA	NA	NA
API Probing Replay Attack 2 (client identifier: cookie)	18	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
API Probing Replay Attack 2 (client identifier: token)	19	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
API Probing Replay Attack 2 (client identifier: IP address)	20	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Data Exfiltration Attack Type 2	21	Tn = [1-42] Tx = [2-43]	Tn = [0-30]	Tn = [1-100]	NA	NA	NA
Excessive Client Connections (client identifier : cookie)	22	Tn = [1-19], Tx = [2-20]	NA	NA	NA	NA	NA
Excessive Client Connections (client identifier : token)	23	Tn = [1-19], Tx = [2-20]	NA	NA	NA	NA	NA
Excessive Client Connections (client identifier : IP address)	24	Tn = [1-19], Tx = [2-20]	NA	NA	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]	NA	NA
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]	NA	NA
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]	NA	NA
Content Scraping Type 2	28	Tn = [1-29] Tx = [2-30]	Tn = [1-100]	NA	NA	NA	NA
Unauthorized client attack (client identifier: IP address)	29	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Single Client Login Attack Type 2 (client identifier: IP address)	30	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
Stolen API Key Attack-API Key	31	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA
Probing Replay Attack - API Key	32	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA	NA	NA
Extended Probing Replay Attack - API Key	33	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA	NA	NA
Account Takeover Type 1	34	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA
Account Takeover Type 2	35	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA
Sequence attack	36	Tn = [1-100] Tx = NA	Tn = [1-100] Tx = NA	NA	NA	NA	NA

Header Inspection	37	Tn = [1-48] Tx = [1-50]	Tn = [1-10] Tx = NA	Tn = [1-18] Tx = [1-20]	Tn = [1-100] Tx = NA	NA	NA
<b>WebSocket API</b>							
WS Cookie Attack	50	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
WS Identity Attack	51	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA	NA	NA	NA
WS DoS Attack	53	Tn = [1-100] Tx = "na"	NA	NA	NA	NA	NA
WS Data Exfiltration Attack	54	Tn = [1-100] Tx = "na"	NA	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 makes periodic REST API calls to an ABS engine which returns JSON reports that are used as events. All the connections between the browser and Splunk are either based on secure token or Splunk universal forwarder. Organizations can utilize the attack information to develop and detect any form of patterns to get a holistic view of attacks.

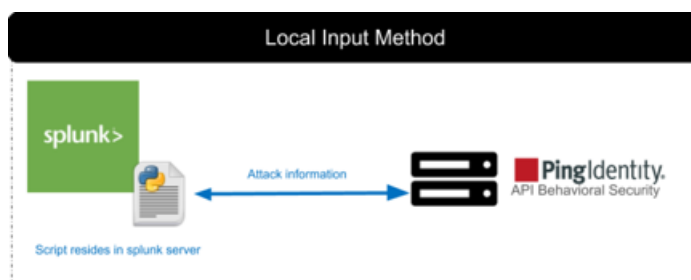
Installing and configuring Splunk for PingIntelligence is a two step process.

1. Install and configure Splunk
2. [Download](#) and configure PingIntelligence ABS splunk script using one of the following two methods:
  - Local input method, or
  - Splunk universal forwarder method

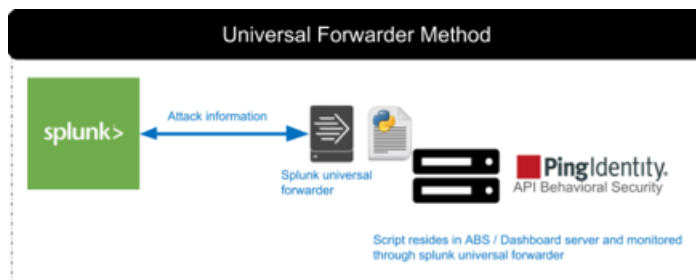
After you have configured the method to send data, the PingIntelligence ABS script creates an individual event based on the attacks reported by ABS and posts it to splunk.

There are two methods through which data is sent to Splunk.

- **Local Input method:** In local input method, a local script is run on the Splunk Enterprise which fetches the attack data from PingIntelligence ABS and sends the information to Splunk using a scripted input method.



- **Splunk Universal Forwarder method:** In this method, the Splunk Universal Forwarder monitors a log directory which contains output of the attack data. The script which provides attack detail runs periodically through a CRON job.



## System requirements

Following are the system requirements to deploy Splunk for PingIntelligence:

- Splunk enterprise 7.1.3 or higher
- `splunkuser` must be configured and all files should be under the ownership of `splunk`.
- Python version:
  - Python 2.7.5-69.e17\_5
  - Python-libs 2.7.5-69.e17\_5
- Define `SPLUNK_HOME` if this local script is run from a Splunk Enterprise node.

## 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	<code>pi_events_source_type</code>
Destination app	Search and Reporting (Can change for your apps)
Category	Structures
Indexed Extractions	<code>json</code>
Timestamp	<code>%a %b %d %H:%M:%S %Y</code>
BREAK_ONLY_BEFORE	<code>(\{)</code>

MUST_BREAK_AFTER	(\})
------------------	------

## Edit Source Type: pi\_events\_source\_type

Description

Used by Pingintelligence scripts to index AB

Destination app

Search &amp;

Category

Stru

Indexed Extractions ?

j

Timestamp

**Advanced**

Name	Value	
CHARSET	UTF-8	×
BREAK_ONLY_BEFORE	(\)	×
INDEXED_EXTRACTIONS	json	×
MUST_BREAK_AFTER	(\)	×
NO_BINARY_CHECK	true	×
SHOULD_LINEMERGE	false	×
TIME_FORMAT	%a %b %d %H:%M:%S %Y	×
category	Structured	×
description	Used by Pingintelligence script	×
disabled	false	×
		×

## Types of data captured

Splunk for PingIntelligence captures the following three types of data:

- Attack type event
- Metrics summary
- Metrics details

**Attack type event:** The attack event captures the components listed in the following table:

Field	Description
access_type	Type of event
api_name	Name of the API
attack_type	The name and type of attack
identifier	The value of attack identifier, for example, cookie, token, or IP address
identifier_count	The number of times a specific identifier was captured
identifier_type	The type of identifier (cookie, token, or IP address)
source_ip	Source IP address of the attack
timestamp	Timestamp of the event

Following is a sample screen shot depicting the attack type event:

i	Time	Event
>	18/03/2019 06:10:01.000	<pre>{ [-]   access_type: attack   api_name: shop   attack_type: Data Exfiltration Attack Type 2   identifier: IbxyhvPyih   identifier_count: 1   identifier_type: cookie   source_ip: [ [-]     100.10.5.81   ]   timestamp: Mon Mar 18 05:40:00:000 2019 }</pre> <p><a href="#">Show as raw text</a></p> <p>host = ip-172-31-19-163.ap-south-1.compute.internal   source = /opt/pin</p>

**Metrics summary:** Each summary event contains an outline of activities occurred in this API for a time period. The summary metrics captures the following:

Fields	Description
access_type	Type of event

api_name	Name of the API
api_url	URL of the API
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 <code>earlier_date</code> 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 <code>later_date</code> would be 14th April, 6 PM.
most_popular_device	Most popular device accessing this API in the specific time period
most_popular_ip	Most popular IPs accessing this API in the specific time period
most_popular_method	Most popular method used to access this API in the specific time period
total_requests	Total number of requests received by this API in the specific time period
no_sessions	Number of sessions connected through IP
sessions	Number of sessions connected through token or cookie
success	Number of successful connections in the specific time period
servers	List of backend servers accessed by this API in the specific time period.
timestamp	Time stamp of this event which in this case is same as later date of this event

Following is a sample screen shot showing the summary metrics event:

```
> 18/03/2019 { [-]
  05:40:00.000  access_type: api_metrics_summary
                  api_name: shop-books
                  api_url: /shopapi-books
                  earlier_date: Mon Mar 18 05:30:00:000 2019
                  later_date: Mon Mar 18 05:40:00:000 2019
                  most_popular_device: LINUX
                  most_popular_ips: [ [-]
                                100.10.2.95
                                100.10.2.100
                                100.10.2.60
                                ]
                  most_popular_method: GET
                  no_sessions: 10515
                  servers: [ [-]
                        { [-]
                          count: 5213
                          server: 10.0.4.5:4200
                        }
                        { [-]
                          count: 5302
                          server: 10.0.4.5:4100
                        }
                        ]
                  sessions: 0
                  success: 9860
                  timestamp: Mon Mar 18 05:40:00:000 2019
                  total_requests: 10515
                }
```

[Show as raw text](#)

host = ip-172-31-19-163.ap-south-1.compute.internal | source



**Metrics details:** Each detail event contains all the activities occurring in this API between earlier date and later date. This is further classified into per session of token, cookie or IP. Metrics details captures the following:

Fields	Description
access_type	Type of event
api_name	Name of the API
device	Device accessing API in this event
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 <code>earlier_date</code> 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 <code>later_date</code> would be 14th April, 6 PM.
path_info	The API path accessed
servers	List of backend servers accessed by this API in the specific time period.
session_id	Session ID name
source_ip	Source IP of this event
timestamp	Time stamp of this event which in this case is same as later date of this event
total_requests	Total number of API requests for this event
user_agent	The user agent used for accessing this API

Following is a sample screen shot showing the detailed metrics event:

```

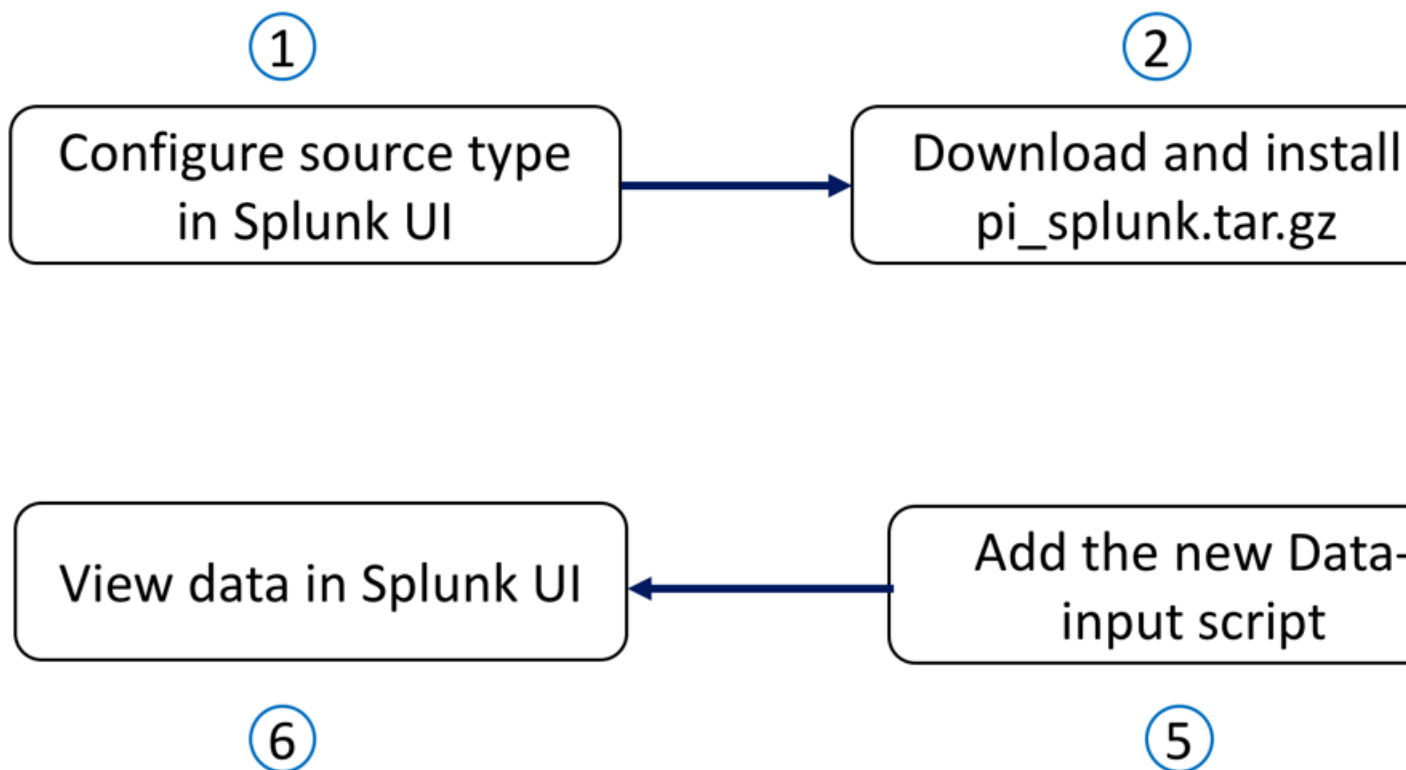
> 18/03/2019 { [-]
05:40:00.000  access_type: api_metrics_detail
                api_name: shop-electronics
                device: [ [+]
                ]
                earlier_date: Mon Mar 18 05:30:00:000 2019
                later_date: Mon Mar 18 05:40:00:000 2019
                path_info: [ [-]
                { [-]
                  count: 4
                  path: /shopapi-electronics/order
                }
                { [+]
                }
                { [+]
                }
                { [+]
                }
                { [+]
                }
                ]
                server: [ [-]
                { [-]
                  count: 22
                  server: 10.0.4.5:4100
                }
                ]
                session_id: fyVLOzIGRi
                source_ip: [ [-]
                { [+]
                }
                ]
                timestamp: Mon Mar 18 05:40:00:000 2019
                total_requests: 22
                user_agent: [ [-]
                { [-]
                  count: 22
                  user_agent: MOZILLA
                }
                ]
                }

```

## Local Input Method installation and configuration

About this task

The installation and configuration process of Local Input Method is depicted in the diagram below:



### Steps

#### 1. Download and install pi-splunk-4.0.tar.gz

After completing the prerequisite step, download the pi-splunk-4.0.tar.gz file from the [download](#) site and extract it to scripts folder in Splunk Enterprise Server. At the Splunk Enterprise command line, enter the following command:

```
tar -xvf pi-splunk-4.0.tar.gz -C $SPLUNK_HOME/bin/scripts/
```

```
root@splunk_enterprise:~#> tar -xvf pi-splunk-4.0.tar.gz -C $SPLUNK_HOME/
bin/scripts/
pingidentity/
pingidentity/splunk/
pingidentity/splunk/bin/
pingidentity/splunk/config/
pingidentity/splunk/logs/
pingidentity/splunk/data/
pingidentity/splunk/data/pi_events_data
pingidentity/splunk/logs/pi_events.log
pingidentity/splunk/config/pi_events.properties
pingidentity/splunk/bin/pi_events.py
```

The following table provides details of the directory structure after you untar the Splunk script:

Directory name	Contents
----------------	----------

bin	▪ pi_events.py: The script to be run from Splunk GUI.
config	Contains pi_events.properties
data	Contains pi_events.data
logs	Contains pi_events.log

2. Copy the pi\_events.py script to \$SPLUNK\_HOME/bin/scripts/ and change the permissions.

a. Copy pi\_events.py script to \$SPLUNK\_HOME/bin/scripts/

```
root@splunk_enterprise:~# cp
  $SPLUNK_HOME/bin/scripts/pingidentity/splunk/bin/pi_events.py
  $SPLUNK_HOME/bin/scripts/pi_events.py
```

**Note:** Splunk UI accepts script present only at \$SPLUNK\_HOME/bin/scripts/ directory

b. Change permissions of the script to splunk user

```
root@splunk_enterprise:~# chown -R splunk. $SPLUNK_HOME/bin/scripts/
  pi_events.py
root@splunk_enterprise:~# chown splunk. $SPLUNK_HOME/bin/scripts/
  pingidentity
```

3. Configure pi\_events.properties file with ABS IP

```
[default]
# Dashboard properties file
# ABS Hostname/IPv4 address
abs.host=< Hostname / IPv4 address >

# ABS REST API port
abs.port=8080

# ABS access key
abs.access_key=<ABS Access Key>

# ABS secret key
abs.secret_key=<ABS Secret Key>

# ABS query offset (seconds. default value 1800 seconds)
abs.query.offset=1800

# ABS query window (seconds. default value 600 seconds)
abs.query.window=600

# Splunk log (path of splunk log)
logfile=pi_events.log
```

The following table provides a description of the pi\_events.properties file variables.

Entry	Description
abs.host	The hostname or IPv4 address of ABS host
abs.port	The management port of ABS for REST API communication. The default port is 8080.

<code>abs.access_key</code>	The abs access keys configured in <code>abs_init.js</code> file during installation. You can also get these details in <code>auth_info</code> collection in <code>abs_metadata</code> in MongoDB.
<code>abs.secret_key</code>	The abs secret keys configured in <code>abs_init.js</code> file during installation. You can also get these details in <code>auth_info</code> collection in <code>abs_metadata</code> in MongoDB.
<code>abs.query.offset</code>	The time in past for which the script window fetches data. The value is specified in seconds. Recommended value is 1800 seconds.  <b>Example:</b> If the current time is 10 AM and you have set an offset of 1800 secs (30-minutes) with a query window ( <code>abs.query.window</code> ) of 600 secs (10-minutes), then the query time would be from 9:20 AM to 9:30 AM.
<code>abs.query.window</code>	The query window is the time interval in seconds for which the script fetches the data from ABS. The minimum and recommended value is 600-seconds.
<code>logfile</code>	The log file name.

4. Configure `pi_events.py` script from UI to run at periodic intervals: **Splunk Enterprise# Settings# Data Inputs# Local Inputs# Scripts# Add new**

**Add Data**

Select Source    Input S...

**Files & Directories**  
Upload a file, index a local file, or monitor an entire directory.

**HTTP Event Collector**  
Configure tokens that clients can use to send data over HTTP or HTTPS.

**TCP / UDP**  
Configure the Splunk platform to listen on a network port.

**Scripts** >  
Get data from any API, service, or database with a script.

Configure  
Scripted in  
Learn More

Source

**Note:** The interval is set to 10-minutes (600-seconds) as shown the in the screen shot above.


# Input Settings

Optionally set additional input parameters for this data input as follows:


## Source type

The source type is one of the default fields that the Splunk platform assigns to all incoming data. It tells the Splunk platform what kind of data you've got, so that the Splunk platform can format the data intelligently during indexing. And it's a way to categorize your data, so that you can search it easily.

## App context

Application contexts are folders within a Splunk platform instance that contain configurations for a specific use case or domain of data. App contexts improve manageability of input and source type definitions. The Splunk platform loads all app contexts based on precedence rules. [Learn More](#) 

## Host

When the Splunk platform indexes data, each event receives a "host" value. The host value should be the name of the machine from which the event originates. The type of input you choose determines the available configuration options. [Learn More](#) 

## Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your

- **Source type:** The script output is JSON. Select the source type as JSON from Structured sub-category.
  - **App Context:** Use the search and reporting as Search
  - **Host:** Provide hostname of the ABS server
  - **Index:** Create an index name, for example, `pi_events`.
5. Complete the configuration. Verify whether events are flowing into splunk search app.

List ▾

✎ Format

&lt; Hide Fields

☰ All Fields

**SELECTED FIELDS***a* host 1*a* source 2*a* sourcetype 2**INTERESTING FIELDS***a* access\_type 1*a* api\_name 3*a* attack\_type 10*a* identifier 100+

# identifier\_count 1

*a* identifier\_type 2*a* index 1

# linecount 3

*a* punct 7*a* source\_ip[] 100+*a* splunk\_server 1*a* timestamp 40**i**

Time

&gt;

12/4/18

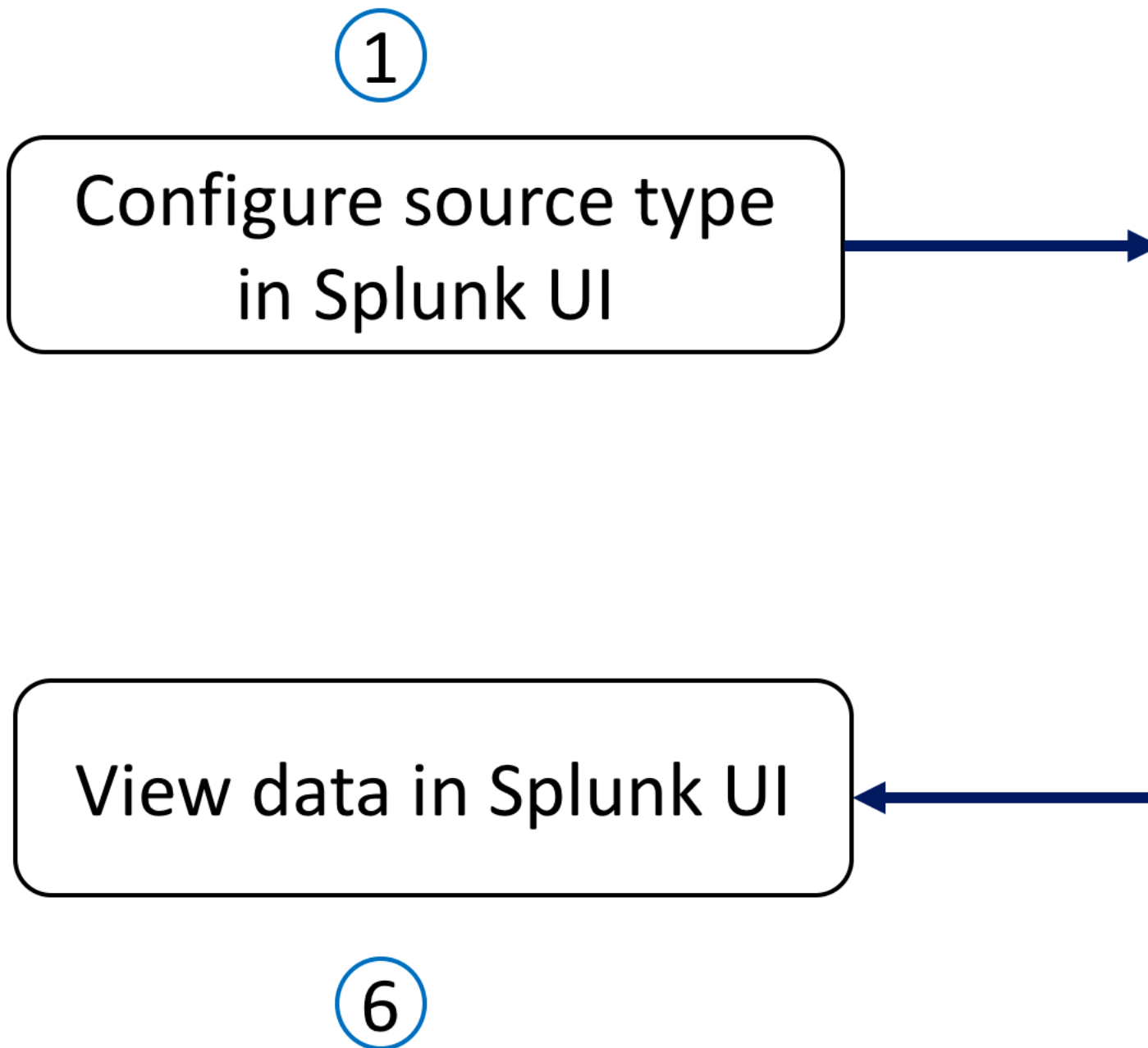
12:29:47.000 AM



## 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. Download the `pi_splunk.tar.gz` file from the [download](#) site and extract it to `/opt` directory:

```
root@pi_nodes:pi_dir:#> tar -xvf pi-splunk-4.0.tar.gz -C /opt/  
pingidentity/  
pingidentity/splunk/  
pingidentity/splunk/bin/
```

```

pingidentity/splunk/config/
pingidentity/splunk/logs/
pingidentity/splunk/data/
pingidentity/splunk/data/pi_events_data
pingidentity/splunk/logs/pi_events.log
pingidentity/splunk/config/pi_events.properties
pingidentity/splunk/bin/pi_events.py

```

The following table provides details of the directory structure after you untar the Splunk script:

Directory name	Contents
bin	pi_events.py: The script to be run from Splunk GUI.
config	Contains pi_events.properties
data	Contains pi_events.data
logs	Contains pi_events.log

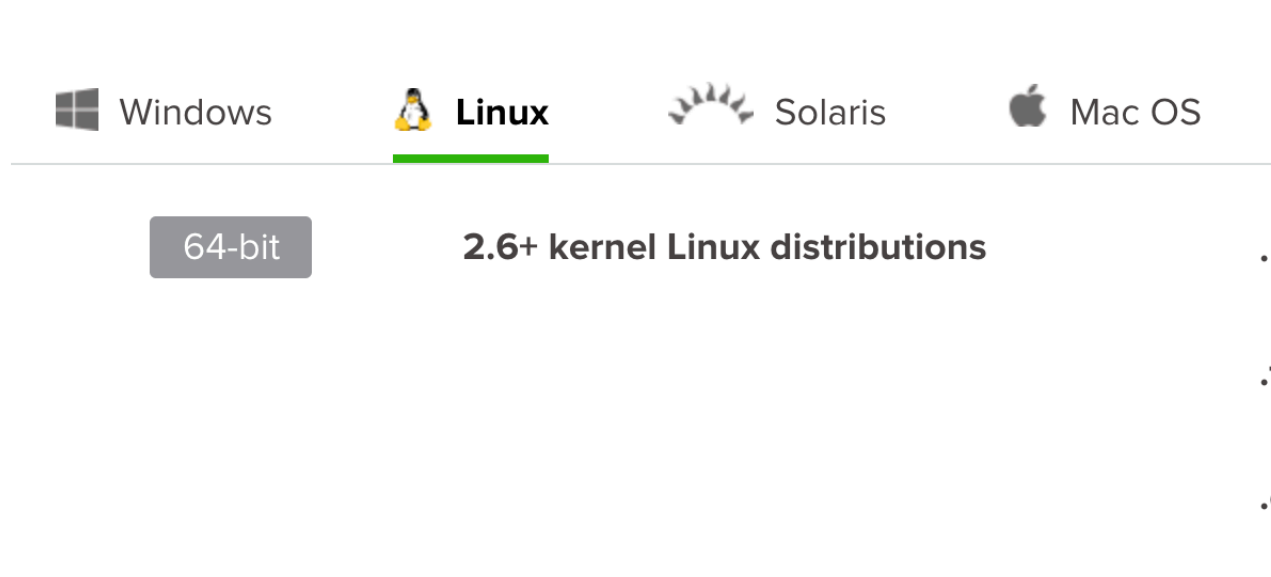
**Note:** Details of pi\_events.log, pi\_events.properties and pi\_events.py are provided in Local input method.

2. Install and configure Splunk Universal forwarder and start the instance using following steps:
  - a. Download Splunk Universal Forwarder

## Splunk Universal Forwarder 7.2.4

Universal Forwarders provide reliable, secure data collection from remote systems and data consolidation. They can scale to tens of thousands of remote systems, collecting data from a wide variety of sources.

### Choose Your Installation Package



- b. Install the Splunk universal forwarder by entering the following command:

```
[root@ABS]# tar -xvf splunkforwarder-7.2.0-8c86330ac18-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

```
[root@ABS]# ./splunk add forward-server ip:port
Splunk username: admin
Password:
Added forwarding to: 192.168.1.158:9997.
```

- e. Add monitor directory

```
[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@ABS]# cat /opt/splunkforwarder/etc/apps/search/local/inputs.conf
```

```
[monitor:///opt/pingidentity/splunk/data]
index = pi_events
sourcetype=pi_events_source_type
disabled = false
```

#### g. Restart Splunk universal forwarder

```
[root@ABS]# ./splunk restart
```

### 3. Configure pi\_events.properties file with ABS IP

```
[default]
# Dashboard properties file
# ABS Hostname/IPv4 address
abs.host=< Hostname / IPv4 address >

# ABS REST API port
abs.port=8080

# ABS access key
abs.access_key=<ABS Access Key>

# ABS secret key
abs.secret_key=<ABS Secret Key>

# ABS query offset (seconds. default value 1800 seconds)
abs.query.offset=1800

# ABS query window (seconds. default value 600 seconds)
abs.query.window=600

# Splunk log (path of splunk log)
logfile=pi_events.log
```

The following table provides details of the variables of pi\_events.properties file:

Entry	Description
abs.host	The hostname or IPv4 address of ABS host
abs.port	The management port of ABS for REST API communication. The default port is 8080.
abs.access_key	The abs access keys configured in abs_init.js file during installation. You can also get these details in auth_info collection in abs_metadata in MongoDB.
abs.secret_key	The abs secret keys configured in abs_init.js file during installation. You can also get these details in auth_info collection in abs_metadata in MongoDB.
abs.query.offset	The time in past for which the script window fetches data. The value is specified in seconds. Recommended value is 1800 seconds.  <b>Example:</b> If the current time is 10 AM and you have set an offset of 1800 secs (30-minutes) with a query window (abs.query.window) of 600 secs (10-minutes), then the query time would be from 9:20 AM to 9:30 AM.
abs.query.window	The query window is the time interval in seconds for which the script fetches the data from ABS. The recommended value is 600-seconds.

logfile	The log file name.
---------	--------------------


#### 4. Add entry to crontab

##### a. Open crontab:

```
#crontab -e
```

##### b. Add the following line:

```
*/10 * * * * /opt/pingidentity/splunk/bin/pi_events.py -c  
/opt/pingidentity/splunk/config/pi_events.properties >>  
/opt2/pingidentity/splunk/data/pi_events.data
```

 **Note:** Script has to redirect the logs to `pi_events.data`

5. Verify if data is flowing to Splunk

List

&lt; Hide Fields

☰ All Fields

**i**

&gt;

**SELECTED FIELDS***a* host 1*a* source 1*a* sourcetype 1**INTERESTING FIELDS***a* device{}.count 27*a* device{}.device 6*a* index 1

# linecount 1

*a* path\_info{}.count 13*a* path\_info{}.path 24*a* punct 9*a* server{}.count 27*a* server{}.server 2*a* session\_id 100+*a* source\_in{}.count 32

&gt;

 **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 an 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

**Settings**Alert **PingIntelligence for APIs Alert**

Description PingIntelligence for APIs Alert

Search `index="pi_events" sourcetype="pi_events_source_type" access_type="atta`Alert type **Scheduled** Real-time

Run on Cron Schedule ▾

Time Range Last 600 seconds ▶

Cron Expression `*/10 * * * *`e.g. 00 18 \*\*\* (every day at 6PM). [Learn More](#)

Expires 24 hour(s) ▾

**Trigger Conditions**

Trigger alert when Number of Results ▾

is greater than ▾ 0

Trigger **Once** For each resultThrottle ? 

Enter the values as described in the table below:

Value	Description
Description	PingIntelligence for APIs Alert
Search	Search: <code>index="pi_events" sourcetype="pi_events_source_type" access_type="attack"</code>
Alert Type	Scheduled -> <b>Run on Cron Schedule</b>
Cron Expression	<code>*/10 * * * *</code>

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

+ Add Actions ▾

When triggered

Slack

Channel

Message 

```
-----  
$result.attack_type$ has been  
detected on API: $result.api_  
name$  
-----  
More details :  
`$result._raw$`
```

Attachment

Fields

Advanced settings:

Webhook URL

Slack  
sage t  
)  
Enter  
to the  
sage c  
sert te  
the se  
Option  
ment.  
Show  
the se  
Slack  
rated  
wildca  
You ca  
hook U  
send t  
ferent

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 181

### abs.log messages:

Log message	Description
Warn :-Maximum Transaction limit is reached for this month	This message is logged in <code>abs.log</code> when the transaction limit is reached for the allotted license usage. For more information, see <a href="#">ABS License</a> on page 167
Warn :- Attempt to shutdown ABS from 127.0.0.1	This message is logged in <code>abs.log</code> when shutdown of ABS AI engine is initiated.
Warn :- Failed to delete IPs from IOCs - try again	This message is logged in <code>abs.log</code> 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 <code>abs.log</code> 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 <code>abs.log</code> 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 <code>abs.log</code> 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 <code>abs.log</code> when ABS license has expired. For more inforamtion, see <a href="#">ABS License</a> on page 167
Warn :- MongoDB primary node is down	This message is logged in <code>abs.log</code> when a MongoDB connection failure occurs.
Warn :- Stream init-wait interrupted	This message is logged in <code>abs.log</code> 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 <code>abs.log</code> 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 <code>abs.log</code> when shutdown of MongoDB connection was not successful.
Warn :- Error while loading anomaly dictionary from mongo	This message is logged in <code>abs.log</code> when writing of anomalies to data directory fails.
Warn :- Error while closing file handle for stream config	This message is logged in <code>abs.log</code> when an error occurs while closing the streaming configuration file.
Error: exception while parsing license file <code>/opt/pingidentity/abs/config/PingIntelligence.lic</code>	This message is logged in <code>abs.log</code> 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 <code>/opt/pingidentity/abs/config/PingIntelligence.lic</code> is invalid. ABS will shut down now.	This message is logged in <code>abs.log</code> 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 <code>abs.log</code> when your free ABS license expires.
Attempting to initialize abs, but abs is already in <code>&lt;message&gt;</code>	This message is logged in <code>abs.log</code> when another ABS process is already running.

Log message	Description
<p>error while loading <code>abs.properties</code> &lt;Custom run-time message&gt;</p> <p>The various custom error messages could be:</p> <ul style="list-style-type: none"> <li>▪ property &lt;<code>abs_propertie</code>&gt; is missing</li> <li>▪ invalid value for property <code>log_level</code>. Value should be string and member of [ALL,DEBUG,INFO,WARN,ERROR,FATAL,OFF]</li> <li>▪ property <code>management_port</code> is missing</li> <li>▪ invalid value for property <code>management_port</code>, value should be integer and ( <math>\geq 1</math> &amp;&amp; <math>\leq 65535</math> )</li> <li>▪ invalid value for property <code>jks_password</code>, deobfuscation of password failed. Please make sure you are using the correct <code>config/abs_master.key</code> file</li> <li>▪ invalid value for property <code>jks_password</code>, value should be obfuscated using the 'bin/cli.sh -u admin -p &lt;password&gt; obfuscate_keys' command</li> <li>▪ invalid value for property <code>host_ip</code>, value should be string and ipv4 address</li> <li>▪ property <code>enable_emails</code> is missing</li> <li>▪ invalid value for property <code>smtp_host</code> value should be string and should be as per rfc1024 and rfc1123</li> </ul>	<p>This message is logged in <code>abs.log</code> when:</p> <ul style="list-style-type: none"> <li>▪ Error occurs when <code>abs.properties</code> file is not configured with <code>log_level</code> specifications</li> <li>▪ Error occurs when <code>abs.properties</code> file is not configured with <code>management_port</code> specifications</li> </ul>
<p>error while loading <code>abs_resources.properties</code></p>	<p>This message is logged in <code>abs.log</code> when <code>abs_resources.properties</code> doesn't contain values for memory and CPU parameters</p>
<p>error while initializing mongodb replica set connections</p>	<p>This message is logged in <code>abs.log</code> when MongoDB initialization fails and cannot access a read or write client for connections.</p>
<p>error while reading <code>enable_ssl</code> key from mongo master</p>	<p>This message is logged in <code>abs.log</code> when MongoDB client tries to fetch the key from MongoDB collections.</p>
<p>error while reading <code>root_api_attack</code> key from mongo master</p>	<p>This message is logged in <code>abs.log</code> when MongoDB client tries to fetch the key from MongoDB collections.</p>
<p>error while reading <code>/config/abs.properties</code></p>	<p>This message is logged in <code>abs.log</code> while loading and validating the <code>abs.properties</code> file. Check whether file exists and its permission.</p>
<p>invalid value for property <code>jks_password</code>, value should be obfuscated using the 'bin/cli.sh -u admin -p &lt;password&gt; obfuscate_keys' command</p>	<p>This message is logged in <code>abs.log</code> when an error occurs while deobfuscating the <code>jks_password</code> using the <code>master_key</code></p>
<p>error while loading auth keys from metadata db in mongo</p>	<p>This message is logged in <code>abs.log</code> when MongoDB is not accessible.</p>
<p>error while loading restricted user auth keys from metadata db in mongo</p>	<p>This message is logged in <code>abs.log</code> when MongoDB is not accessible.</p>

Log message	Description
Unable to read <abs_root_dir>/config/abs.jks file	This message is logged in <code>abs.log</code> when <code>abs.jks</code> 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 <code>abs.log</code> when there is an issue when ABS starts.
API Behavioral Security stopped	This message is logged in <code>abs.log</code> when ABS is shut down.
MongoDB heartbeat failure	This message is logged in <code>abs.log</code> when ABS is unable to connect to MongoDB primary node.
ABS started successfully	This message is logged in <code>abs.log</code> when ABS starts.

## AAD log messages

The following tables list the critical log messages from `aad.log` file. `aad.log` file is rotated every 24-hours.

Log messages	Description
error - fatal protocol violation	This message is logged in <code>aad.log</code> when there is a HTTP/(S) protocol error while connecting to ABS or ASE.
error - fatal transport error	This message is logged in <code>aad.log</code> when there is an unknown host for ABS or ASE
warn - http request " + <URL> + ", response status: " + <Response status>	This message is logged in <code>aad.log</code> when ABS or ASE returns HTTP status code that is greater than or equal to 300.
error - error while fetching list api from ase	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - no list api output from ase	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - api id empty from list api at index	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.

Log messages	Description
error- error while fetching api info from ase for api	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - no api details from ase for api	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - invalid api detail json from ase for api	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - error while deleting api	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - error while pushing new api	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
error - error while pushing new server to api	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.
no details from abs for discovered api id	This message is logged in <code>aad.log</code> when there is a connectivity issues with ASE or credentials are wrong. Check AAD and ASE version compatibility. The major versions of AAD and ASE should be the same, for example, 3.2.x or 4.x for both.

## PingIntelligence Dashboard

---

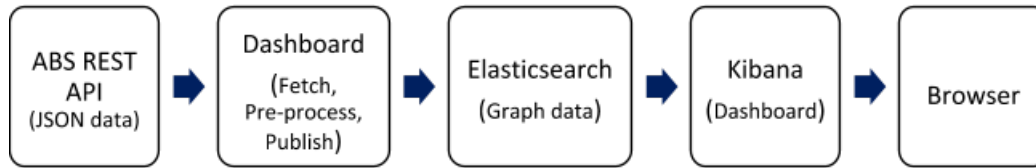
### PingIntelligence for APIs Dashboard

---

PingIntelligence for APIs Dashboard utilizes Elasticsearch and Kibana to provide a graphical view of an API environment including user activity, attack information, and blocked connections. The Dashboard makes periodic REST API calls to an ABS AI engine which returns JSON reports that are used to generate graphs. Organizations can utilize the Dashboard examples to develop direct integration into in-house graphical management systems.

The following chart summarizes the software elements involved and data flow for generating graphs:





For detailed information on installing and configuring an ABS engine, please see the [Manual deployment guide](#).

## System requirements for Dashboard

---

PingIntelligence Dashboard software, Elasticsearch, and Kibana are installed on a separate Linux server (x86\_64 architecture, RHEL 7.6 or Ubuntu 16.04 LTS).

The components can be installed in the following environments:

Component	Environment
Kibana	<b>Browser:</b> IE11+, Chrome, Firefox, and Safari (Mac)
Elasticsearch	RHEL 7.6 or Ubuntu 16.04 LTS
PingIntelligence for APIs Dashboard	RHEL 7.6 or Ubuntu 16.04 LTS

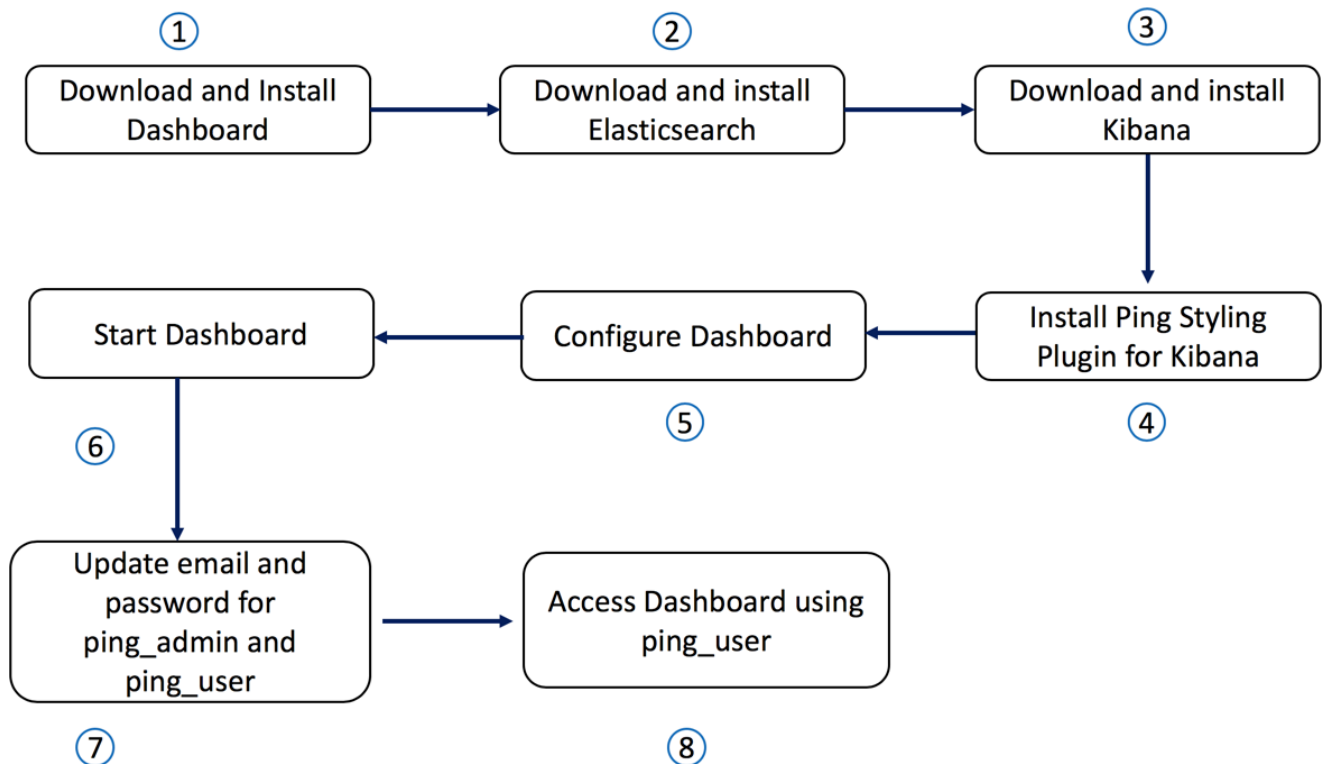
## Dashboard installation

---

Install the following components on a RHEL 7.6 or Ubuntu 16.04 LTS Linux Server:

- Elasticsearch 6.6.0
- Kibana 6.6.0
- OpenJDK 11
- PingIntelligence Dashboard software

The Installation and configuration process of Dashboard is depicted in the diagram below:



Ensure that ports 9200 for Elasticsearch and 443 for Kibana are available for installation. If the ports are not available, then configure different ports for Elasticsearch and Kibana in their respective configuration files. To connect Kibana with Elasticsearch, edit `kibana.yml` and specify the Elasticsearch URL. Here is a snippet from `kibana.yml` file showing the default Elasticsearch URL:

```
# The Elasticsearch instance to use for all your queries.
# elasticsearch.url: "https://localhost:9200"
```

During the installation and configuration process, the following user types are created:

<b>System users</b>	System users manage Elasticsearch and Kibana
<b>Dashboard users</b>	<p><code>ping_admin</code>: Can view and edit Dashboard</p> <p><code>ping_user</code>: Can only view the Dashboard</p>

Create strong passwords of more than six characters for all users. The default password of all user types is `changeme`.

## Prerequisites

Ensure that the following prerequisites are complete before installing PingIntelligence Dashboard software:

1. CA signed SSL certificates for Kibana and Elasticsearch
2. SSL private keys for Kibana and Elasticsearch
3. `wget` and `openssl` must be installed on your system
4. Dashboard, Elasticsearch and Kibana should run as a non-root user

## Install PingIntelligence Dashboard software

Download PingIntelligence Dashboard software from the [download](#) site to a Linux server. Complete the following steps:

1. Start `shell` as a non-root user
2. Change the directory to `/opt`

```
$cd /opt
```

3. Create a `pingidentity` directory

```
$ sudo mkdir pingidentity
```

4. Change the permissions for the `pingidentity` directory. The `pingidentity` directory will be owned by a non-root user.

```
$ sudo chown -R "$(id -nu):$(id -ng)" /opt/pingidentity
```

5. Install Dashboard

```
$ tar -zxf dashboard-4.0.tar.gz
```

The following table shows the directories created when Dashboard is installed:

Directories	Description
<code>bin</code>	ABS Start and Stop scripts; Elasticsearch and Kibana initialization scripts.
<code>config</code>	<code>dashboard.properties</code> file used to configure Dashboard
<code>data</code>	Temporary storage for ABS data
<code>lib</code>	Contains <code>dashboard.jar</code> and dependent external jar files
<code>plugins</code>	Contains the Ping styling plugin
<code>logs</code>	Contains Dashboard log files which are rotated every 24 hours
<code>util</code>	Contains the <code>check_ports_dashboard.sh</code> script to check the availability of default Elasticsearch and ABS ports to connect.

## Download and install Elasticsearch

Complete the following steps to download and install Elasticsearch:

1. Start `shell` as a non-root user
2. Change the directory to `/opt`

```
$cd /opt
```

3. Create an `elasticsearch` directory

```
$ sudo mkdir elasticsearch
```

4. Change the permissions for the `elasticsearch` directory. The `elasticsearch` directory will be owned by a non-root user.

```
$ sudo chown -R "$(id -nu):$(id -ng)" /opt/elasticsearch
```

**5. Change directory to elasticsearch**

```
$ cd /opt/elasticsearch
```

**6. Download Elasticsearch:**

```
$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.8.1.tar.gz
```

**7. Install Elasticsearch:**

```
$ tar -zxvf elasticsearch-6.8.1.tar.gz
```

**8. Change directory:**

```
$ cd /opt/elasticsearch/elasticsearch-6.8.1
```

**Change heap size of Elasticsearch for production deployments:** Complete the following steps:

1. Navigate to `/config/jvm.options` to edit the `jvm.options` file
2. Find the following section in the `jvm.options` file:

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms1g
-Xmx1g
```

3. Change the value of `-Xms1g` to `-Xms4g` and the value of `-Xmx1g` to `-Xmx4g`
4. Save the file

**Configure Elasticsearch**

Configure Elasticsearch by running the `dashboard_elasticsearch_init.sh` script located in the ABS Dashboard `bin` directory. The `dashboard_elasticsearch_init.sh` script asks for the full path of the CA signed certificate. If you do not have a CA signed certificate, generate a self-signed certificate without a passphrase using OpenSSL commands.

```
$ /opt/pingidentity/dashboard/bin/dashboard_elasticsearch_init.sh
```

```
[pingidentity@localhost ~]$ /opt/pingidentity/dashboard/bin/
dashboard_elasticsearch_init.sh
updating elasticsearch configuration
```

```
Enter SSL CA Signed Certificate path >(full path)
Enter SSL Private Key Path >(full path)
```

```
enter pkcs#12 keystore new password >
enter pkcs#12 keystore new password again >
```

```
creating elasticsearch config keystore
config keystore created
```

```
creating password protected pkcs#12 keystore for private key and certificate
pkcs#12 keystore created at config/ssl/elastic-certificates.p12
```

```
Starting Elasticsearch to update default passwords. Please wait for 15
seconds.
```

```
Elasticsearch started with pid 2532 and listening at https://localhost:9200
```

```
updating default user passwords
```

```
## elastic [superuser] password. Remember this password for the Dashboard
setup
enter elastic user new password >
enter elastic user password again >
password updated for user elastic

## kibana [kibana user] password. Remember this password for the Kibana
setup
enter kibana user new password >
enter kibana user password again >
password updated for user kibana

Elasticsearch configuration is complete. Elasticsearch is running at
https://localhost:9200
[pingidentity@localhost ~]$
```

## Install Kibana

Complete the following steps to download and install Kibana:

1. Start **shell** as a non-root user
2. Change the directory to `/opt`

```
$cd /opt
```

3. Create a kibana directory

```
$ sudo mkdir kibana
```

4. Change the permissions for the `kibana` directory to ownership by a non-root user.

```
$ sudo chown -R "$(id -nu):$(id -ng)" /opt/kibana
```

5. Change directory to kibana

```
$ cd /opt/kibana
```

6. Download Kibana:

```
$ wget "https://artifacts.elastic.co/downloads/kibana/kibana-6.8.1-linux-x86_64.tar.gz"
```

7. Install Kibana:

```
$ tar -zxvf kibana-6.8.1-linux-x86_64.tar.gz
```

8. Change directory:

```
$ cd /opt/kibana/kibana-6.8.1-linux-x86_64
```

**Note:** By default, the Kibana uses port 443 with `su/sudo` access. If you want to use any other port, for example 5601, use:

```
$ export KIBANA_DEFAULT_PORT=5601
```

If you are a non-root user, use ports greater 1024.

**Initialize Kibana:** After installing Kibana, initialize Kibana by running the following command:

```
$ /opt/pingidentity/dashboard/bin/dashboard_kibana_init.sh
```

```
[pingidentity@localhost ~]$ /opt/pingidentity/dashboard/bin/
dashboard_kibana_init.sh
updating Kibana configuration
Enter SSL CA Signed Certificate path >(full path)
Enter SSL Private Key Path >(full path)
enter kibana [kibana user] password >
enter kibana [kibana user] password again >
Kibana configuration is complete.
Starting Kibana in the background...
Kibana started with pid 2535 and listening at https://[0.0.0.0]
```

## Install Ping styling plugin

Install the Ping styling plugin for Kibana by entering the following command:

```
./bin/kibana-plugin install
file:///opt/pingidentity/dashboard/plugins/pingstyling-4.0.zip
```

## Configure Dashboard

To configure the Dashboard, edit the `dashboard.properties` file which is located in the `config` directory created when PingIntelligence Dashboard software was installed. In the `dashboard.properties` file, set the `elasticsearch` password to match the password used when [configuring Elasticsearch](#).

```
# 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
+FKmL5cNalDrfVuyc5hIYjimqh7Rnf3bv9hW0+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=6.6.0


### 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

```

Configure all parameters in the `dashboard.properties` file:

Parameter	Description
<b>ABS</b>	
<code>abs.host</code>	IP address of the ABS server
	<div style="border: 1px solid gray; padding: 5px;"> <p> <b>Note:</b> Two options exist to choose an ABS server: 1) Utilize an existing ABS server. 2) For production deployments, Ping Identity recommends dedicating an ABS node exclusively for the Dashboard.</p> </div>
<code>abs.port</code>	REST API port number of the ABS host – See <code>abs.properties</code> Default value is 8080
<code>abs.ssl</code>	Setting the value to true ensures SSL communication between ABS and PingIntelligence for APIs Dashboard
<code>abs.restricted_user</code>	When set to <code>true</code> , Elasticsearch uses the restricted user header (configured in <code>pingidentity/abs/mongo/abs_init.js</code> file) to fetch the obfuscated values of OAuth token, cookie and API keys. When set to <code>false</code> , the admin user header is used to fetch the data in plain text. For more information on admin and restricted user header, see <a href="#">ABS users for API reports</a>
<code>abs.access_key</code>	Access key from ABS – See <code>pingidentity/abs/mongo/abs_init.js</code> . Make sure to enter the access key based on the value set in the previous variable. For example, if <code>abs.restricted_user</code> is set to <code>true</code> , then enter the access key for restricted user. If <code>abs.restricted_user</code> is set to <code>false</code> , then use the access key for the admin user.

abs.secret_key	Secret key from ABS – See <code>pingidentity/abs/mongo/abs_init.js</code> . Make sure to enter the secret key based on the value set in the previous variable. For example, if <code>abs.restricted_user</code> is set to <code>true</code> , then enter the secret key for restricted user. If <code>abs.restricted_user</code> is set to <code>false</code> , 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.
<b>UI</b>	
publish.ui.enable	Set it to <code>true</code> to display PingIntelligence Dashboard. The Dashboard displays attack and metrics data. Set it to <code>false</code> , 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 6.6.0
dashboard.log.level	Log level for Dashboard Default log level is <code>INFO</code> . Another log level is <code>DEBUG</code>
<b>Log4j</b>	
publish.log4j2.enable	Set it to <code>true</code> to send attack data to syslog server. Set it to <code>false</code> to disable sending attack data to syslog server.
<p> <b>Note:</b> Dashboard and Syslog cannot be disabled together.</p>	
log4j2.config	The log4j2 config file which logs the attack data.
log4j2.log.level	Log level for log4j. Default log level is <code>INFO</code> .
log4j2.dependencies.dir	The directory for any log4j configuration dependency. Make sure that it is a directory.

## Update the admin password

PingIntelligence Dashboard software ships with a default username (`admin`) and the default password (`admin`). You can change the default password by using the `update_password` Dashboard CLI command:

```
/opt/pingidentity/dashboard/bin/cli.sh -u admin update_password -p
Password>

New Password>
Re-enter New Password>
Success. Password updated for CLI
```



## Obfuscate keys and passwords

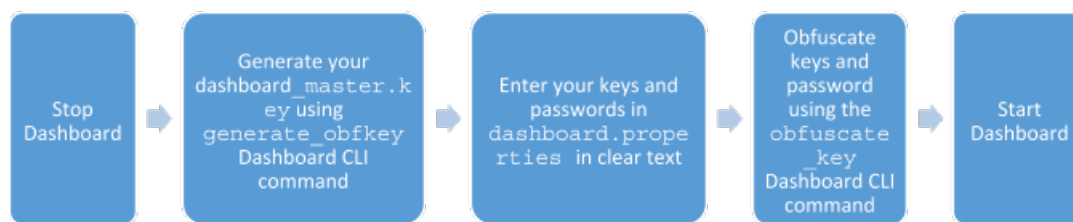
Using the PingIntelligence Dashboard command line interface, you can obfuscate the keys and passwords configured in `dashboard.properties`. The following keys and passwords are obfuscated:

- `abs.access_key`
- `abs.secret_key`
- `es.password`

Dashboard ships with a default `dashboard_master.key` which is used to obfuscate the keys and passwords. It is recommended to generate your own `dashboard_master.key`.

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

The following diagram summarizes the obfuscation process:



### Generate dashboard\_master.key

You can generate the `dashboard_master.key` by running the **generate\_obfkey** command in the Dashboard CLI:

```

/opt/pingidentity/dashboard/bin/cli.sh generate_obfkey -u admin -p
Password>

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

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

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

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

creating new obfuscation master key
Success: created new obfuscation master key at /opt/pingidentity/dashboard/
config/dashboard_master.key
  
```

### Obfuscate key and passwords

Enter the keys and passwords in clear text in `dashboard.properties` file. Run the **obfuscate\_keys** command to obfuscate keys and passwords:

```

/opt/pingidentity/dashboard/bin/cli.sh obfuscate_keys -u admin -p
Password>

Please take a backup of config/dashboard.properties before proceeding

Enter clear text keys and password before obfuscation.
  
```

```

Following keys will be obfuscated
config/dashboard.properties: abs.access_key, abs.secret_key and es.password

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

obfuscating /opt/pingidentity/dashboard/config/dashboard.properties

Success: secret keys in /opt/pingidentity/dashboard/config/
dashboard.properties obfuscated

```

[Start Dashboard](#) after passwords are obfuscated.

**i Important:** After the keys and passwords are obfuscated and the Dashboard has started, move the `dashboard_master.key` to a secure location away from the Dashboard for security reasons. Before restarting the Dashboard, the `dashboard_master.key` must be present in the `/opt/pingidentity/dashboard/config/` directory.

## Start Dashboard

---

### Prerequisite:

For the PingIntelligence Dashboard to start, the `dashboard_master.key` must be present in the `/opt/pingidentity/dashboard/config` directory. If you have moved the master key to a secured location, copy it to the `config` directory before executing the `start` script.

To start the PingIntelligence Dashboard, navigate to the `/opt/pingidentity/dashboard/bin` directory and enter the following command:

```
[pingidentity@localhost bin]# ./start.sh
```

```
[pingidentity@localhost bin]# ./start.sh
Dashboard 4.0 starting..
Please see /opt/pingidentity/dashboard/logs/dashboard.log for more details
[pingidentity@localhost bin]#
```

After the PingIntelligence Dashboard has started, wait for 15 seconds for Dashboard to create the following two users:

- `ping_admin`
- `ping_user`

**i Note:** Immediately after starting PingIntelligence Dashboard, [change the password](#) for both the users. The default password for both the user is `changeme`.

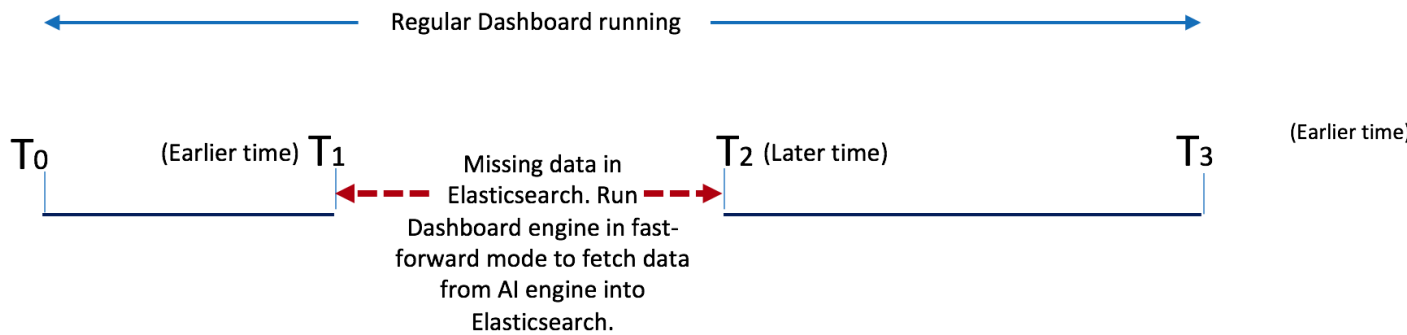
## Dashboard fast forward

---

You can start PingIntelligence Dashboard in a 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.
- You have a fresh installation of Dashboard after a certain period of AI engine installation and you want to populate the Dashboard with data from when ABS AI engine was first started.

The following diagrams summarize the use case for Dashboard's fast-forward mode:



### Missing data in Elasticsearch

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.

**i 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
<code>dashboard.fastforward.earlier_time</code>	The query start date and time in <code>YYYY-DD-MMTHH:mm</code> format.
<code>dashboard.fastforward.later_time</code>	The query end date and time in <code>YYYY-DD-MMTHH:mm</code> format.
<code>dashboard.fastforward.query.range</code>	The time in minutes that Dashboard queries the AI engine in a single pass.
<code>dashboard.fastforward.query.cooling_period</code>	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 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 4.0.1

```

## Syslog for attack data

PingIntelligence Dashboard also 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 version 2.11.2 to publish attack data to syslog.

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` file in the Dashboard config directory. The config file supports other formats available with Log4j including `.properties`, `.json`, or `.yaml`. 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:

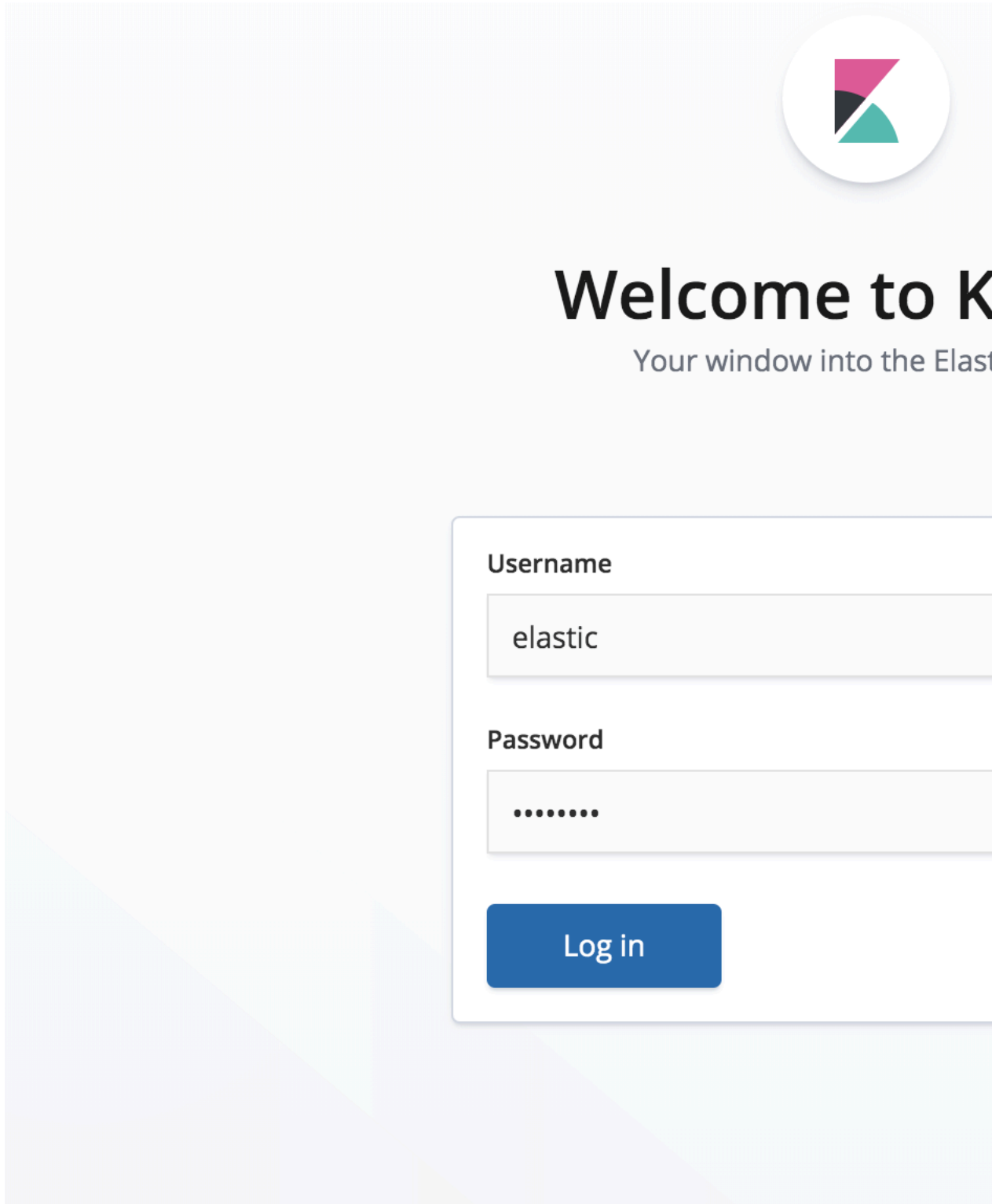
```
<!-- ### Syslog RFC5424 format, TCP -->
  <Syslog name="TCP_RFC5424"
    host="localhost"
    port="614"
    appName="APIIntelligence"
    id="App"
    enterpriseNumber="18060"
    facility="LOCAL0"
    messageId="Audit"
    format="RFC5424"
    newLine="true"
    protocol="TCP"
    ignoreExceptions="false"
    mdcId="mdc" immediateFail="false" immediateFlush="true"
    connectTimeoutMillis="30000" reconnectionDelayMillis="5000"/>
```

## Access the PingIntelligence Dashboard

---

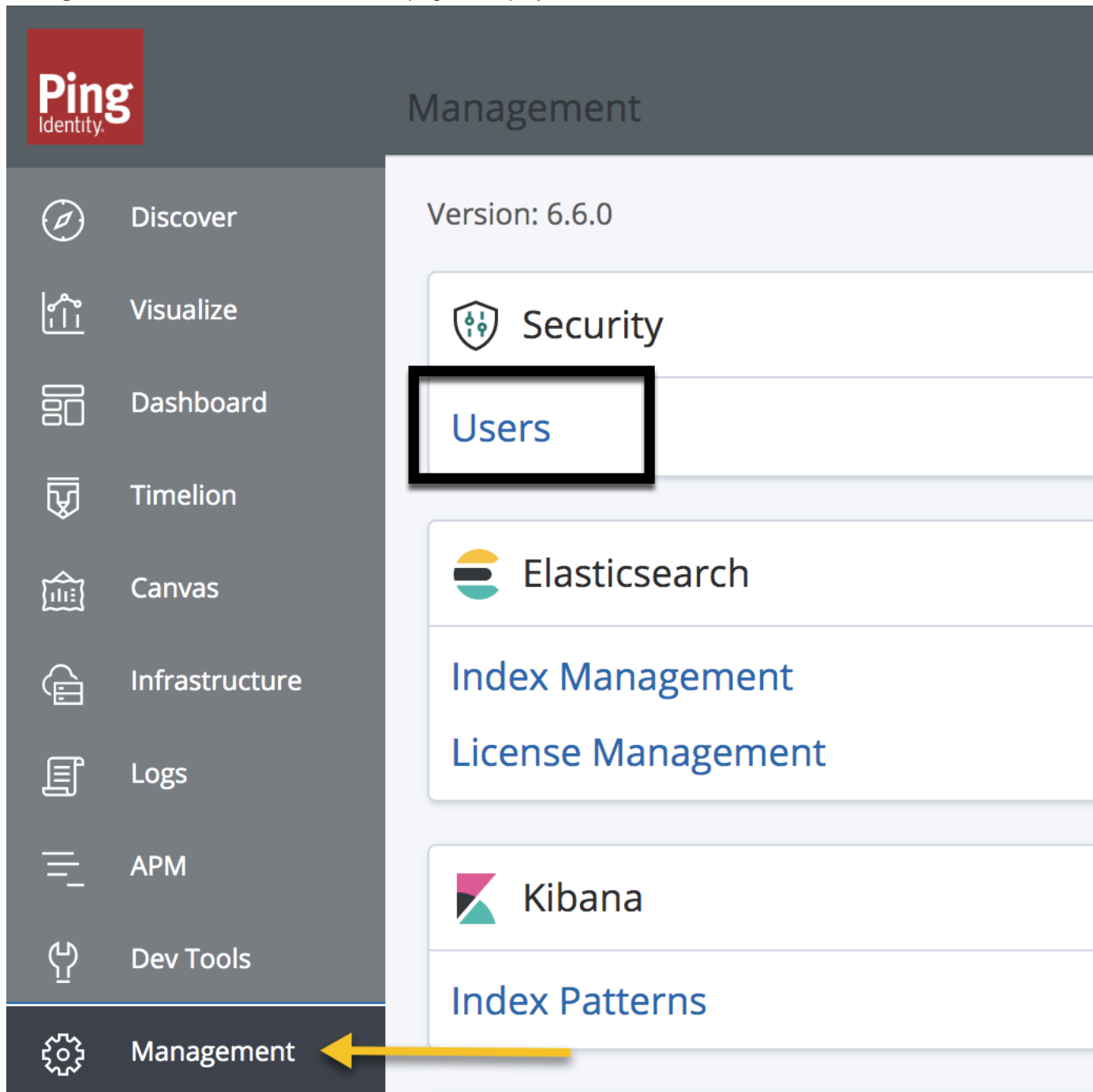
Access the main dashboard with a browser at this URL: <https://<ip:port>/>. In the URL, `<ip:port>` is the IP address and port configured in `kibana.yml`. The default port is 443. Change the password of the two users `ping_admin` and `ping_user` by completing the following steps:

1. Navigate to the PingIntelligence Dashboard URL and log in using `elastic` user and the password set during [Elasticsearch configuration](#). The Kibana landing page is



displayed.

2. In the Kibana landing page, click **Management**. The Management page is displayed. In the **Management** tab, click **Users**. The Users page is displayed:



3. On the Users page, click on **ping\_admin** to change the email and password of ping\_admin user.

The screenshot shows the Ping Identity Management Security interface. The top navigation bar includes 'Management' and 'Security', with 'Users' and 'Roles' as sub-sections. The left sidebar contains a navigation menu with icons and labels for 'Discover', 'Visualize', 'Dashboard', 'Timelion', 'Canvas', and 'Infrastructure'. The main content area is titled 'Users' and features a search bar with the placeholder text 'Search...'. Below the search bar, there is a list of users with checkboxes for selection. The first user is 'Full Name' with an upward arrow indicating sorting. The other two users are 'Ping Dashboard Admin' and 'Ping Dashboard User'.

<input type="checkbox"/>	Full Name ↑
<input type="checkbox"/>	Ping Dashboard Admin
<input type="checkbox"/>	Ping Dashboard User



4. On the ping\_admin Users page, update the **Email** and **Password** fields and click **Save**:

**Edit "ping\_admin" user**

**Username**  
ping\_admin  
Username's cannot be changed after creation.

**Full name**  
Ping Dashboard Admin

**Email address**  
admin@index  
A valid email address is required

**Roles**  
ping\_admin\_role ×

**Password**  
.....

**Confirm password**  
.....

Save password Cancel

Update user Cancel Delete user

Repeat steps 2 through 4 for ping\_user to update **Email** and **Password**. Then log in with ping\_user credentials to view the dashboard. Here is a partial screen grab of the main dashboard:



> Search... (e.g. status:200 AND extension:PHP)



Dashboard

Add a filter +

82

IP Blacklist

18

Cookie Blacklist

## Attacks

API ⚡

URL ⚡

shop-books

/shopapi-books

shop-electronics

/shopapi-electronics

shop

/shopapi

decoy

/decoy

The main dashboard provides the following information:

- **Attack Summary:** total number of attacks, number of IP addresses, cookies, tokens, API keys, and usernames on the blacklist. Note: a single IP could generate more than one attack, so the sum of the blacklist counts may be less than the total number of attacks.
- **Time series chart of attacks:** total number of attacks on each API over time
- **Total number of attacks on each API**
- **API Metrics:** Activity generated on each API - Requests Accepted (green) and Requests Rejected (blue).
- **API Information:** information on each API including:
  - **Type** – regular or decoy (see ASE Admin Guide for decoy API explanation)
  - **Protocol** – HTTP, WebSocket
  - **URL** – URL to access API
  - **Hostname** – host name for the API.
  - **Servers** – number of servers hosting the API

For each API, an API-specific Dashboard can be displayed by clicking on the API name in the main dashboard.

The screenshot shows the PingIdentity dashboard interface. At the top left is the Ping Identity logo. The main header contains 'Dashboard' and 'PingIntelligence for APIs Dashboard'. A search bar is located below the header with the text '>\_ Search... (e.g. status:200 AND extension:PHP)'. A sidebar on the left contains a 'Dashboard' menu item. The main content area features two summary cards: '82 IP Blacklist' and '18 Cookie Blacklist'. Below these is a section titled 'Attacks' which contains a table of API endpoints. A yellow arrow points from the sidebar to the 'shop-books' entry in the table.

API	URL
shop-books	/shopapi-books
shop-electronics	/shopapi-electronics
shop	/shopapi
decoy	/decoy

The following screenshot shows an API specific dashboard:



>\_ | Search... (e.g. status:200 AND extension:PHP)



Dashboard

Add a filter +

[← Main Dashboard](#)

## API Attacks

Attack	Count
Data Exfiltration Attack	3,426
Extreme App Activity	2,882
Extreme Client Activity	906
Data Poisoning Attack	589
API Probing Replay	431
Decoy Attack	417

Export: [Raw](#) [Formatted](#)

1

DoS/DDoS Attacks

If graphs are not displayed due to Kibana errors, refresh your browser. Each dashboard displays the following API specific reports:


**Attack reporting:**

- **Attack summary:** total number of attacks, number of IP addresses, cookies, tokens, API keys, and usernames on the blacklist. Note: a single IP or cookie could generate more than one attack, so the sum of the blacklist counts may be less than the total number of attacks.
- **Attack types:** count of each type of attack. Attack type examples include data exfiltration, stolen cookies, etc. See ABS Admin Guide for a complete list of attacks

**API metric reporting**

- **Requests/API URLs** – number of requests on each valid API URL
- **Requests/Device-Type** – number of requests per device type

**Server error codes** – Count of each error code returned from API servers.

 **Note:** The graphs displayed are reference Kibana graphs. You can create scripts and graphs that suit your deployment using REST API calls to the ABS engine.

## View blacklist

You can view the total number of entries on each blacklist from the main dashboard. Click on the count on any blacklist type (e.g. IP

The screenshot shows the Ping Identity dashboard. The top navigation bar includes the Ping Identity logo, a search bar with the text '> Search... (e.g. status:200 AND extension)', and the text 'Dashboard PingIntelligence for APIs Dashbo'. Below the navigation bar is a sidebar with a 'Dashboard' link and a menu icon. The main content area is titled 'Attacks' and features a table with two columns: 'API' and 'URL'. The table lists several API endpoints and their corresponding URLs. Above the table, there are two large blue numbers: '82' for 'IP Blacklist' and '18' for 'Cookie Blacklist'. Two yellow arrows point upwards from the 'Attacks' section towards these numbers.

API	URL
shop-books	/shopapi-books
shop-electronics	/shopapi-electronics
shop	/shopapi
decoy	/decoy

Clicking on any of the blacklist, displays the list of client identifiers on each black list type.

## Dashboard PingIntelligence for API Dashboard: Blacklist

>\_ | Search... (e.g. status:200 AND extension:PHP)

Add a filter +

[← Main Dashboard](#)

Active User Blacklist

Active Token Blacklist

**Token** ⌵

e9f8bff5f7782f1dc3042a1ac

e56e97600690a5b892f83488

b93a9bfb94b5fab0e29ab38f

99a0141e1335c0a47cda9d3-

6c53cb6025b2c8d5ecab1b4-

307f0e2c2c8d989214fdc33a

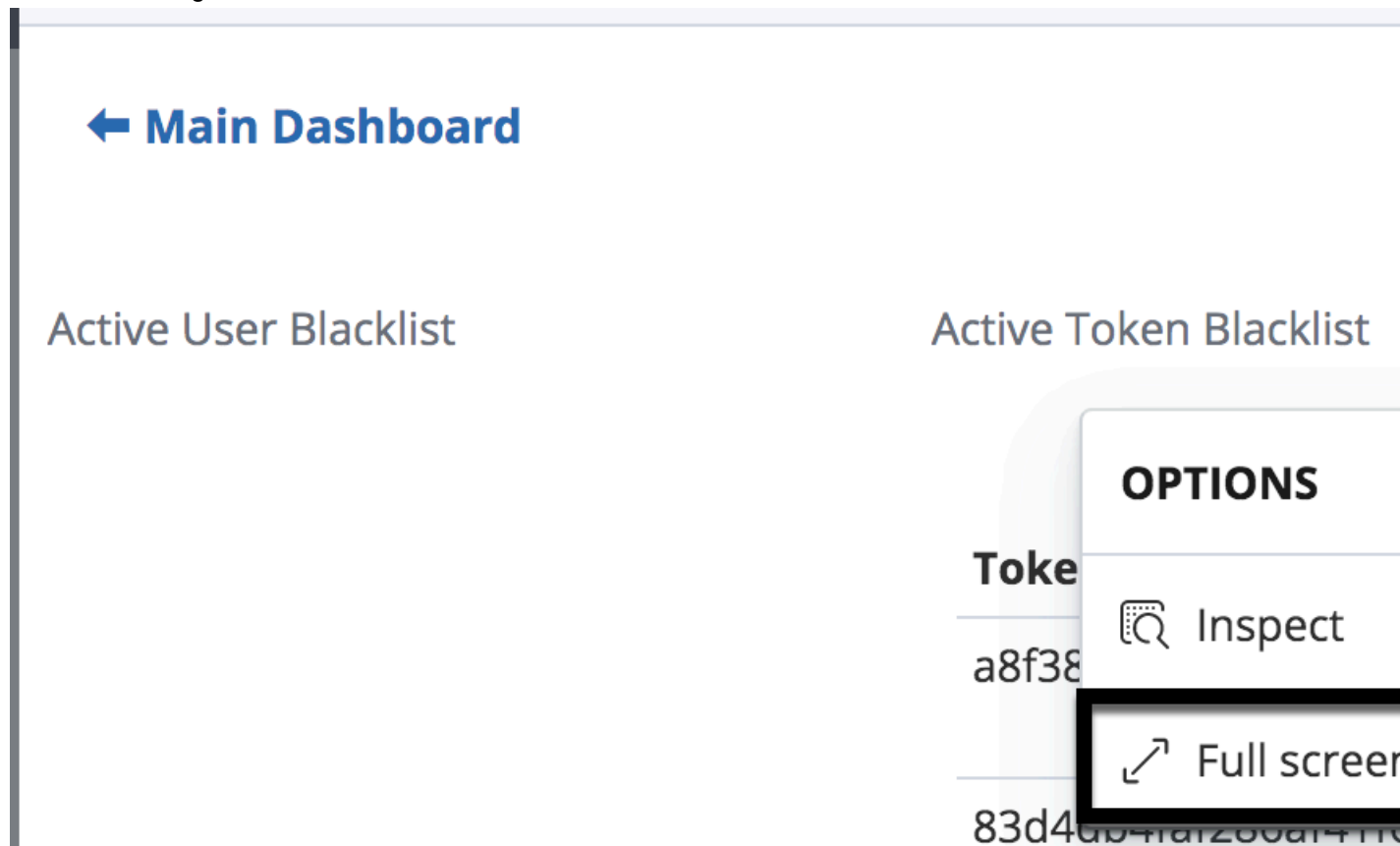
088ee704fe6c360f259c5cdc2

05b349cbfb7b2212a0dc7b9-

No results found



**Expanding the blacklist view:** You can expand a single the blacklist by hovering the mouse over the blacklist, and clicking on the three-dots, and then clicking on **Full Screen** as shown in the screenshot below:



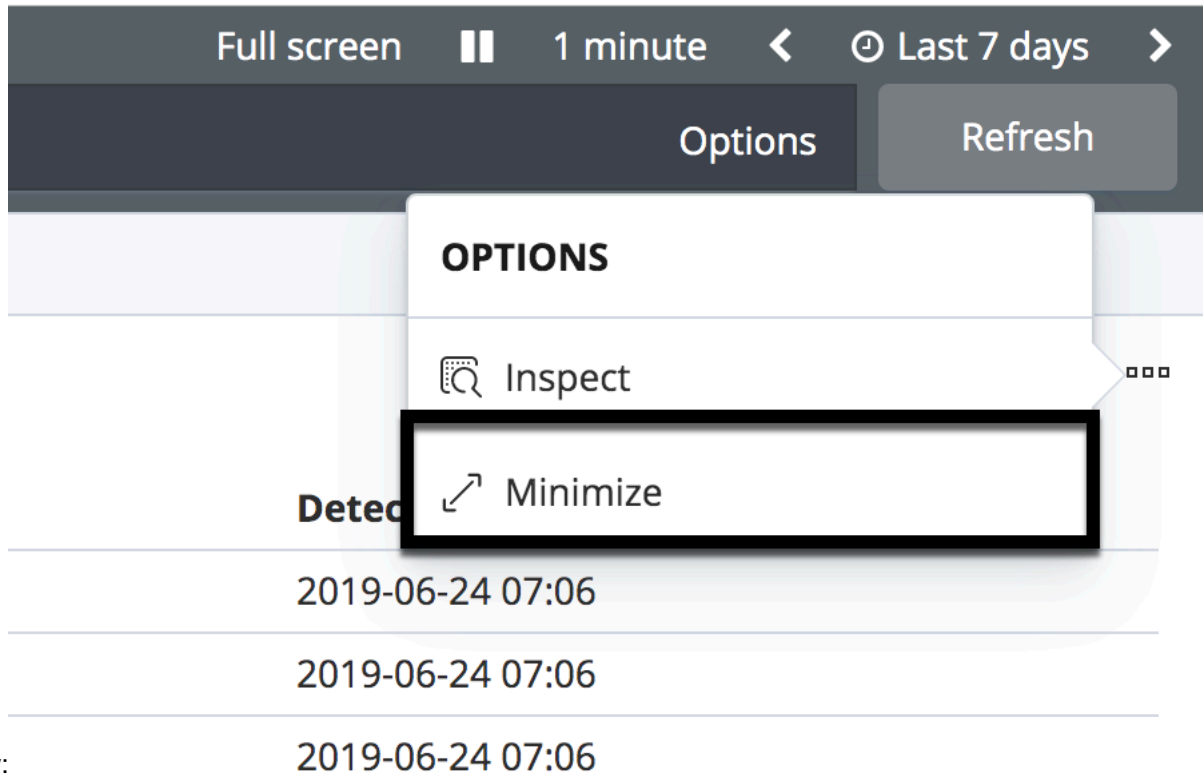
The following screenshot shows the expanded blacklist:

The screenshot displays the Ping Identity API Dashboard. The top navigation bar includes the Ping Identity logo, the text "Dashboard", and "PingIntelligence for API Dashboard: Bla". A search bar is present with the placeholder text ">\_ Search... (e.g. status:200 AND extension:PHP)".

The main content area is titled "Active Token Blacklist" and features a section header "Token" with a dropdown arrow. Below this, a list of 20 hexadecimal tokens is displayed, each on a separate line and separated by horizontal dividers. The tokens are:

- e9f8bff5f7782f1dc3042a1ac7b64b5541713b74a6c5
- e56e97600690a5b892f834885fa2ed8691aaa52965
- b93a9bfb94b5fab0e29ab38ffe1a2d939d41df7cca1
- 99a0141e1335c0a47cda9d349de4d662d67cf260d8
- 6c53cb6025b2c8d5ecab1b4a96464899427efa4624
- 307f0e2c2c8d989214fdc33ae63dd3e40cc9addbd7
- 088ee704fe6c360f259c5cdc2d7ada3f8a1976c90f7e
- 05b349cbfb7b2212a0dc7b982ea413610b58c8d9a4
- ea4862136973910b288f339aaa5b216de059bf742
- c147b5d5196a8dc50555c09f1b2204a42213ab93c9
- 9f4662363c4405dd2871d4ff80a807614ec93bff8e8
- 35ab09d56fdec409fbf0ff99c856ed9ba1a75f7e0cb
- 1d001f949bb11728c49dc8a085b01e7a5ed2366c64
- 134ed716eaeccca5ede1521e504a7f911df5c24bb
- b1f9f15f6d59e838777027f469f3509a072ed933987
- a8f380795cd7140b1d82e0629b3603c896bbc7f824
- 83d4db4faf286af41f608189ef5061f61fd29a44ce05

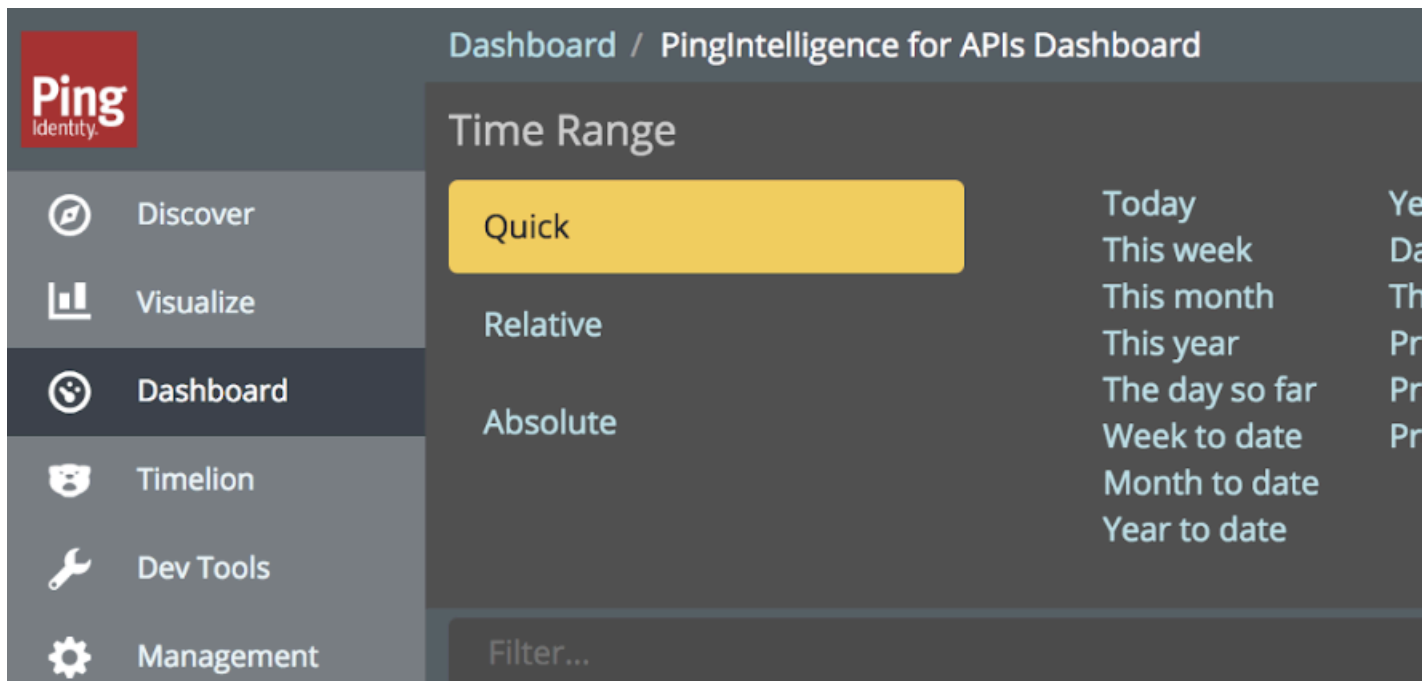
To navigate back to the blacklist view, click on the three-dots and then click on **Minimize** as shown in the screenshot



below:

### Dashboard time series

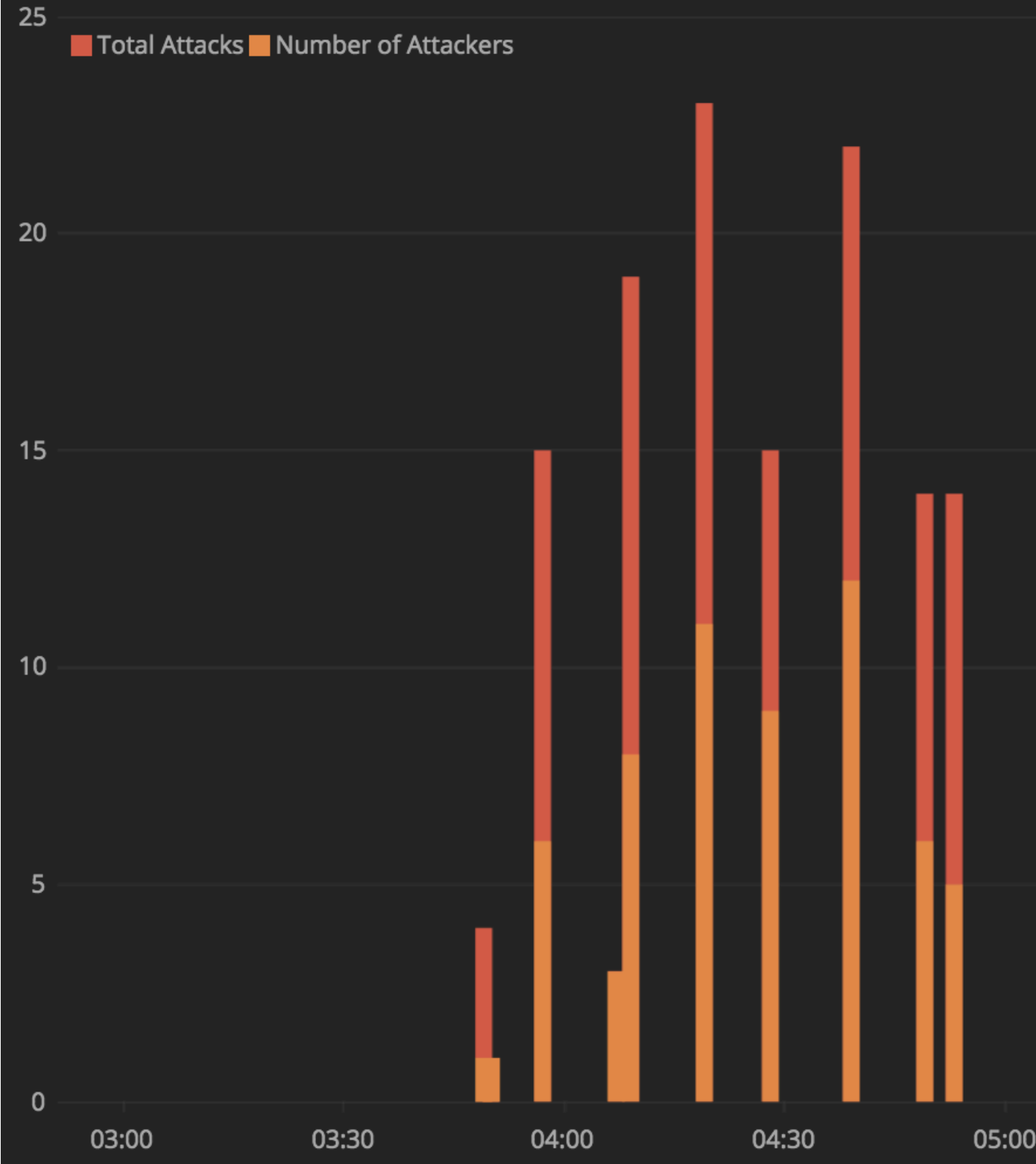
PingIntelligence Dashboard shows the attacks in a time series format. Change the time series duration by clicking on the time-period arrow located on the top right corner of the dashboard. See the example in the following screen capture:



Here is a screen capture of the time series data:

### Global Attacks Time Series

#### Attacks



## Stop Dashboard

To stop ABS Dashboard, navigate to the `/opt/pingidentity/dashboard/bin` directory and enter the following command:

```
[piuser@localhost bin]# ./stop.sh
```

```
[piuser@localhost bin]# ./stop.sh
Dashboard 4.0 stopping...
Dashboard is stopped
```

## 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.

**i Warning:** When the purge script is run, all files are permanently deleted from the Elasticsearch data directory.

Run the purge script, on the Dashboard command line:

```
/opt/pingidentity/util/purge_elasticsearch.sh -d 3
```

In the above example, `purge.sh` deletes all files older than 3 days. Here is a sample output:

```
/opt/pingidentity/util/purge_elasticsearch -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
```

When you use the `-X DELETE` option, the system goes back to a fresh installation state.

## Purge dashboard logs

A The `purge.sh` script either archives or purges removes processed access log files which are stored in the `/opt/pingidentity/dashboard/logs` directory.

**i 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. 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 2 -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 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/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.

## 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 <code>dashboard.log</code> when there is a HTTP/(S) protocol error while connecting to ABS or Elasticsearch.
error - fatal transport error	This message is logged in <code>dashboard.log</code> when there is an unknown host for ABS or Elasticsearch.
error - error while sending message to syslog	This message is logged in <code>dashboard.log</code> when the syslog server is not reachable, or there is an error in configuration of SSL or non-SSL connections.

Log messages	Description
error - capacity full in syslog consumer worker, retries exhausted, ignoring this message	This message is logged in <code>dashboard.log</code> 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 <code>dashboard.log</code> when ABS, or Elasticsearch socket is not closed properly.
error - error while flushing file stream	This message is logged in <code>dashboard.log</code> 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 <code>dashboard.log</code> 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 <code>dashboard.log</code> 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 <code>dashboard.log</code> 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 <code>dashboard.log</code> 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 <code>dashboard.log</code> when ABS or Elasticsearch returns HTTP status code that is greater than or equal to 300.
Dashboard stopped	This message is logged in <code>dashboard.log</code> when Dashboard is shutdown.

## PingIntelligence Deployment

---

### Automated deployment

---

#### Automated 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
- Automated API Definition (AAD) tool

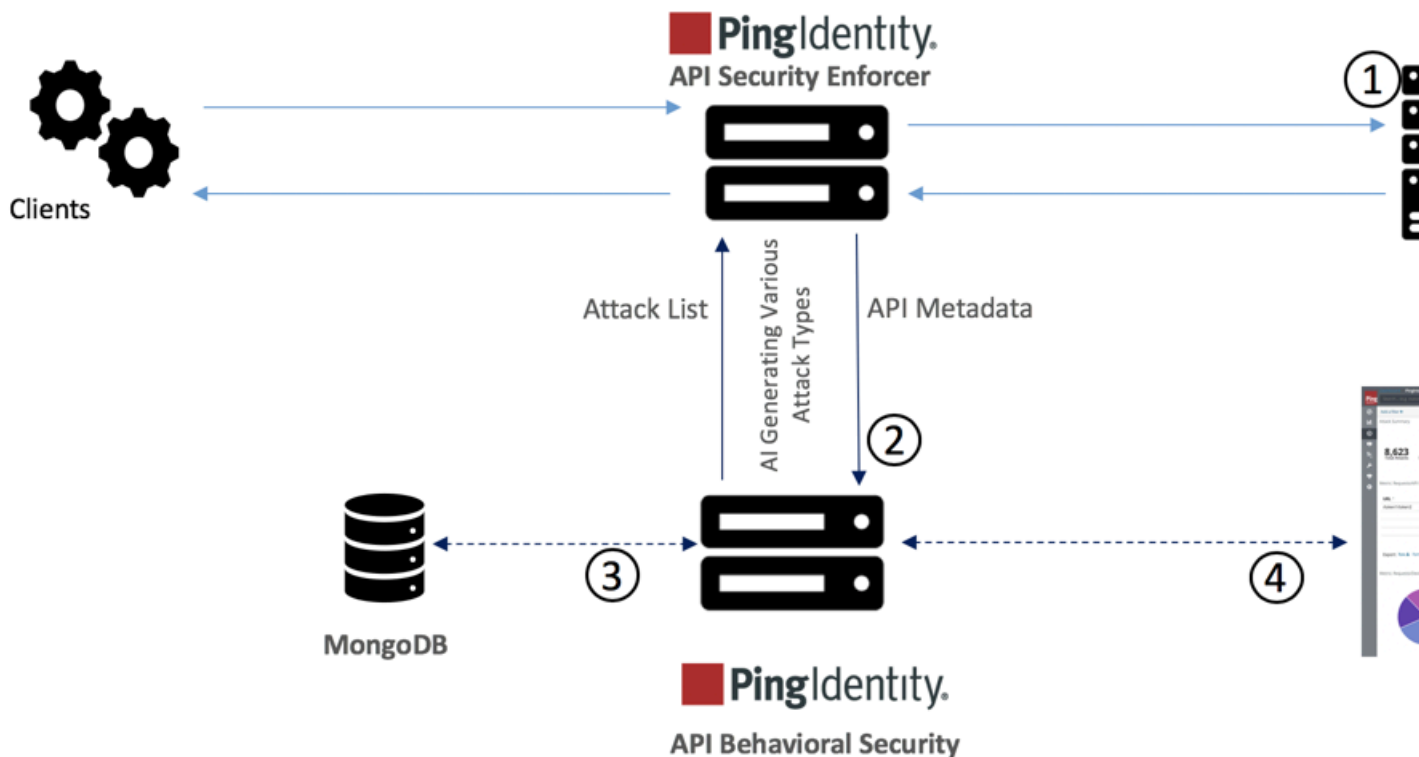




## ASE deployment modes

### Inline mode

In the inline deployment mode, ASE sits at the edge of your network to receive the API traffic. It can also be deployed behind an existing load balancer such as AWS ELB. In inline mode, 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.



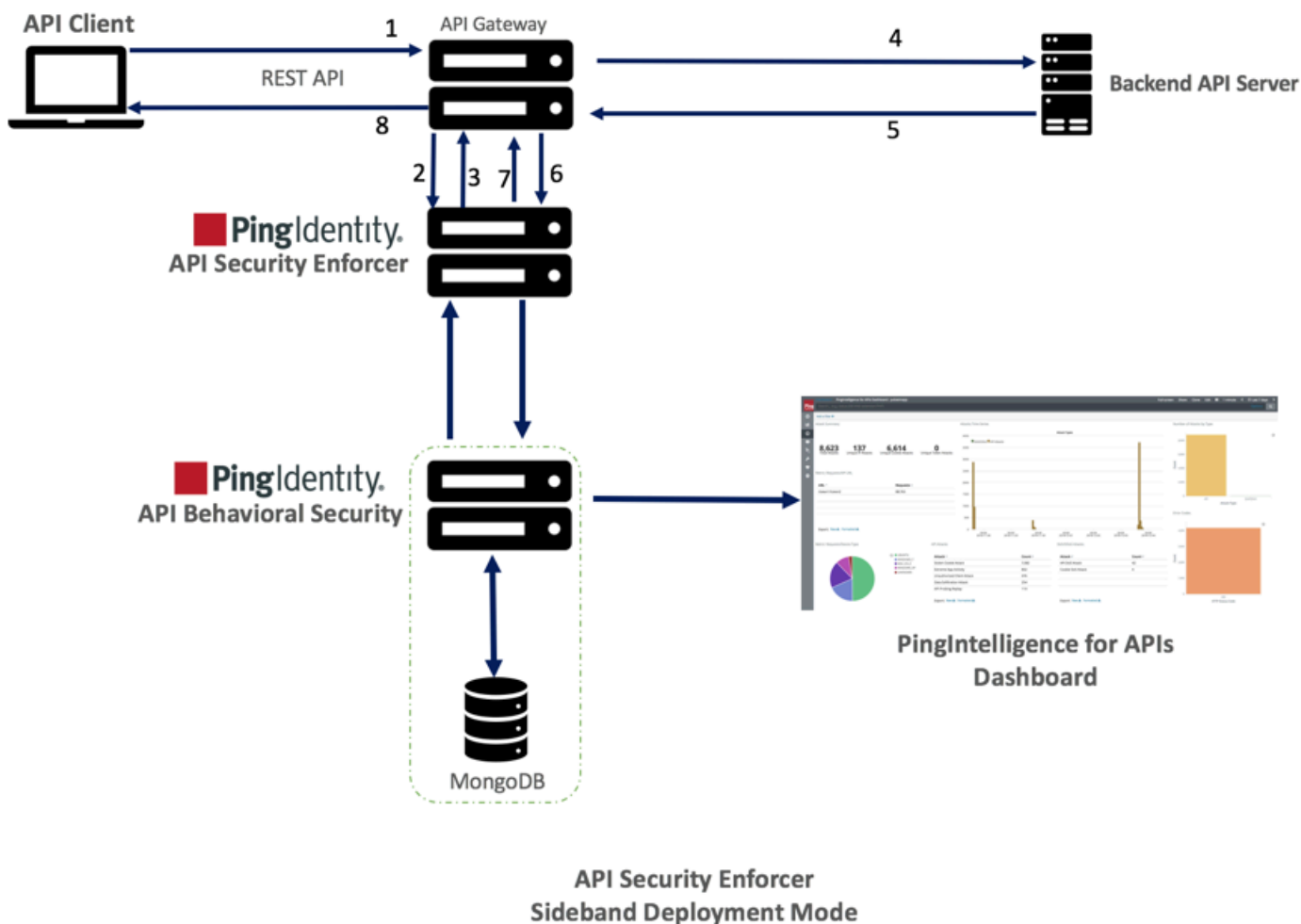
### API Security Enforcer Inline Deployment Mode

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

### 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 sideband mode, set the **mode=sideband** in the `ase-defaults.yml` file.



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](#).

## Step 1: Setup host system

Setup the management machine

The deployment framework requires RHEL 7.6 machine for running the deployment. The machines on which PingIntelligence components are installed can be either a RHEL 7.6 or Ubuntu 16.04 LTS. Provision a management machine which is used to run the PingIntelligence deployment.

### Steps for all the machines

Complete the following steps on all the provisioned machines.

**i Important:**

1. Make sure that the deployment is homogenous with respect to the provisioned machines. Either all the machines should be running RHEL 7.6 or Ubuntu 16.04 LTS. Do not create a set up having both types of machines.
2. All the 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.
3. If you are deploying the setup on a Ubuntu machine, make sure that the MongoDB machine has `libcurl4-openssl-dev`.

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

```
#useradd pi-user
```

2. Change the password

```
#passwd pi-user
```

3. **i 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 pi-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
```

### Steps for management machine:

1. Login to 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-4.0.2.tar.gz
```

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

Directory	Description
<code>ansible</code>	Contains the different <code>yml</code> files

bin	Contains the <code>start.sh</code> and <code>stop.sh</code> scripts. Do not edit the contents of this directory.
certs	<p>Contains ASE, ABS, Elasticsearch, Kibana, and MongoDB self-signed certificates and keys. Elasticsearch and Kibana certificates are in the dashboard directory.</p> <p><b>Note:</b> 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 <code>certs</code> directory. Make sure that the keys are password-less.</p>
config	Contains the default settings file for ASE, ABS, Dashboard, and AAD. This file is 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 <code>external</code> directory.
keys	After the installation is complete, the master keys of all the products are saved here.
license	Contains <code>ase</code> and <code>abs</code> directories that have the ASE and ABS license file.
logs	Contains the log files for automated installation
software	<p>Contains the binary files for PingIntelligence components:</p> <ul style="list-style-type: none"> <li>▪ ASE</li> <li>▪ ABS</li> <li>▪ Dashboard</li> <li>▪ AAD</li> <li>▪ AMT</li> </ul> <p>The directory also contains <code>updated_packages</code> sub-directory which stores the PingIntelligence updated binaries with new master keys. You can use these binaries for future use.</p>

### Configure password-less authentication

PingIntelligence requires 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/authorized_keys` file on all the machines where PingIntelligence components are installed.

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

### Configure 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`.

- **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=4.0
IssueDate=2019-06-01
EnforcementType=0
ExpirationDate=2019-12-30
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=4.0
IssueDate=2019-06-01
EnforcementType=0
ExpirationDate=2019-12-30
```

```
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
```

## Configure hosts file

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.

1. Navigate to the `pi-api-deployment/config` directory and edit the `hosts` file to add the IP address of all the machines.
2. **Configure the hosts file**

Configure the following fields in the config/hosts file:

Variable	Description
IP addresses <ul style="list-style-type: none"> <li>▪ [ase]</li> <li>▪ [abs]</li> <li>▪ [mongodb]</li> <li>▪ [dashboard]</li> <li>▪ [elasticsearch]</li> <li>▪ [kibana]</li> <li>▪ [abs_reporting_node]</li> <li>▪ [aad_amt]</li> </ul>	Configure the following IP addresses: <ul style="list-style-type: none"> <li>▪ ASE IP address</li> <li>▪ ABS IP address</li> <li>▪ MongoDB IP address and port</li> <li>▪ ABS reporting node IP address</li> <li>▪ Dashboard IP address</li> <li>▪ Kibana IP address</li> <li>▪ Elasticsearch IP address</li> <li>▪ AAD and AMT</li> </ul> <p>If you want to install all the components on a single machine for testing purpose, give the same IP address for all the component. If you want a distributed deployment, provide different IP addresses.</p> <p><b>Important:</b> 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. It is a good practice to keep reporting node and dashboard node separate.</p>
mongo_port	Port number for MongoDB. Default value is 27017. The port number provided in this variable is considered only if MongoDB port number is not provided along with IP address. If you have provided port number with MongoDB IP address, the value in this variable is ignored.

installation_path	<p>Configure the path where you would want the PingIntelligence components to be installed. The default value is <code>/home/pi-user</code>.</p> <p><b>i Important:</b> The path that you provide in the <code>installation_path</code> 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.</p>
install_with_sudo	<p>When set to <code>false</code>, the script installs PingIntelligence for a normal user. When set to <code>true</code>, the script installs PingIntelligence as a root user if the port number of ports configured are less than 1024.</p>
install_as_service	<p>Set it to <code>true</code> if you want to install PingIntelligence components as a service. To install PingIntelligence components, you must be a root user. Set <code>install_with_sudo</code> as <code>true</code>.</p> <p><b>i Note:</b> If you do not have <code>root</code> access at the time of running automated deployment, you can install PingIntelligence components as a service manually. For more information, see</p> <p>If you install PingIntelligence components as a service, the components are automatically restarted when the system is rebooted. Check the <code>ansible.log</code> file to verify starting PingIntelligence components as a service.</p>
install_mongo	<p>Set it to <code>true</code> if you want automated deployment to install MongoDB. Set it to <code>false</code> if you want to use an existing MongoDB installation. Default value is <code>true</code>.</p> <p><b>i Important:</b> Configure the MongoDB IP address and port number even if <code>install_mongo</code> is set to <code>false</code>. MongoDB details are required to configure <code>abs.properties</code> file.</p>
jdk11_download_url	<p>The automated script requires OpenJDK 11.0.2.</p> <p><b>i Note:</b> If your machine does not have internet access, then download the OpenJDK 11.0.2 and save the file as <code>openjdk11.tar.gz</code> in external directory.</p>

mongodb_download_url	<p>MongoDB download URL. A default URL is populated in the <code>hosts</code> file.</p> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>The default URL is RHEL version of MongoDB. If you are installing on Ubuntu, configure the MongoDB Ubuntu download URL.</li> <li>If your machine does not have internet access, then download the MongoDB 4.2.0 and save the file as <code>mongodb.tgz</code> in <code>externaldirectory</code>.</li> </ol>
elasticsearch_download_url	<p>Elasticsearch download URL. A default URL is populated in the <code>hosts</code> file.</p> <p><b>Note:</b> If your machine does not have internet access, then download the Elasticsearch 6.8.1 and save the file as <code>elasticsearch-6.8.1.tar.gz</code> in <code>externaldirectory</code>.</p>
kibana_download_url	<p>Kibana download URL. A default URL is populated in the <code>hosts</code> file.</p> <p><b>Note:</b> If your machine does not have internet access, then download the Kibana 6.8.1 and save the file as <code>kibana-6.8.1-linux-x86_64.tar.gz</code> in <code>externaldirectory</code>.</p>
ansible_ssh_user	<p>Ansible <code>ssh</code> user. The default value is <code>pi-user</code>.</p>

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

```
[ase]
192.168.11.148

[elasticsearch]
192.168.11.149

[dashboard]
192.168.11.149

[kibana]
192.168.11.149

[abs]
192.168.11.145

[abs_reporting_node]
192.168.11.147

[mongodb]
192.168.11.146 mongo_port=27017

[aad_amt]
192.168.11.144

[all:vars]
```



```

mongo_port=27017

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

# Configure install_with_sudo to true if the any of the ports used for
ASE, ABS
# and Dashboard is <1024. That component will be started using sudo.
install_with_sudo=false

# install_as_service set to true will start ase, abs, aad, dashboard,
elasticsearch
# and kibana as systemd services.
install_as_service=false

# 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

# 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 with password-less access
ansible_ssh_user=pi-user

```

### Manually download third-party components

If your Ping Management VM does not have internet access then you can download the software using the steps mentioned below. Download the individual components and save the file in the `external` directory.

1. Install Ansible version 2.6.2 on the Ping Management host. The Ping Management VM is the VM from which the automated deployment script is run to deploy the various PingIntelligence software.
2. 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](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](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](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`.

**Note:** If you are planning to use Dashboard only for syslog messages and not the GUI, do not download Elasticsearch and Kibana. Configure the `dashboard-defaults.yml` to choose the type of Dashboard installation.

## 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
- Automated API Definition tool
- Attack Management tool

**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
  dashboard-4.0.2.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 159M Jun 07 00:01 abs-4.0.2.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 38M Jun 07 00:01 ase-
  rhel-4.0.2.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 6M Jun 07 00:01 aad-4.0.2-tar.gz
-rw-r--r--. 1 pingidentity pingidentity 5M Jun 07 00:01 pi-
  amt-4.0.2.tar.gz
```

At the end of automated deployment, the master keys of all the products are deleted from the individual instances for security reasons. The master keys are available in the `keys` directory on the management instance. Starting of any PingIntelligence component requires a master key to be present. To manually restart any of the PingIntelligence components, copy the corresponding master key back to the `config` directory.

For more information on the master key of each component see the respective *Admin Guides*.

## Change default settings

The deployment package provides `yml` files to change the default settings of ASE, ABS, Dashboard, 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](#)
- [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
<code>mode</code>	Sets the mode in which ASE is deployed. The default value is <code>inline</code> . Set the value to <code>sideband</code> if you want ASE to work in the sideband mode.

<code>http_ws_port</code>	Data port used for HTTP or WebSocket protocol. The default value is 8090.
<code>https_wss_port</code>	Data port used for HTTPS or secure WebSocket protocol. The default value is 8443.
<code>management_port</code>	Management port used for CLI and REST API management. The default value is 8010.
<code>cluster_manager_port</code>	ASE node uses this port number to communicate with other ASE nodes in the cluster. The default value is 8020.
<code>keystore_password</code>	The password for ASE keystore. The default password is <code>asekeystore</code> .
<code>cluster_secret_key</code>	This key is used for authentication among ASE cluster node. All the nodes of the cluster must have the same <code>cluster_secret_key</code> . 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 <code>yourclusterkey</code> .
<code>enable_sideband_keepalive</code>	This key is used only in ASE sideband mode. Setting it to <code>true</code> , ASE sends a keep-alive in response header for the TCP connection between API gateway and ASE. With the default <code>false</code> value, ASE sends a connection close in response header for connection between API gateway and ASE.
Email default settings	Configure the following settings: <ul style="list-style-type: none"> <li>▪ <code>enable_emails</code>: Set it to <code>true</code> for ASE to send email notifications. Default value is <code>false</code>.</li> <li>▪ <code>smtp_host</code> and <code>smtp_port</code></li> <li>▪ <code>sender_email</code>: Email address used from which email alerts and reports are sent.</li> <li>▪ <code>email_password</code>: Password of sender's email account.</li> <li>▪ <code>receiver_email</code>: Email address at which the email alerts and reports are sent.</li> </ul>
CLI admin password	The default value for CLI admin is <code>admin</code> . To change the password, you need to know the current password. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> Do not change the <code>current_admin_password</code>. The automated deployment uses the current password to reset the new password.</p> </div>

**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
```

### 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
<code>management_port</code>	Port for ABS to ASE and REST API to ABS communication. The default value is 8080.
<code>log_port</code>	Port for ASE to send log files to ABS. The default value is 9090.
<code>mongo_username</code> and <code>mongo_password</code>	MongoDB user name and password. The default user name is <code>absuser</code> and the default password is <code>abs123</code> .
<code>mongo_cache_size</code>	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 <code>true</code> if MongoDB is configured to use SSL connections. The default value is <code>false</code> .  PingIntelligence deployment ships with a default self-signed certificate.
access_key and secret_key	The access key and secret for the admin user. For more information on different ABS users, see <a href="#">ABS users</a>
access_key_ru and secret_key_ru	The access key and secret for the restricted user. For more information on different ABS users, see <a href="#">ABS users</a>
jks_password	The password of the JKS Keystore. The default password is <code>abs123</code> .
Email default settings	Configure the following settings: <ul style="list-style-type: none"> <li>▪ <code>enable_emails</code>: Set it to <code>true</code> for ASE to send email notifications. Default value is <code>false</code>.</li> <li>▪ <code>smtp_host</code> and <code>smtp_port</code></li> <li>▪ <code>sender_email</code>: Email address used from which email alerts and reports are sent.</li> <li>▪ <code>email_password</code>: Password of sender's email account.</li> <li>▪ <code>receiver_email</code>: Email address at which the email alerts and reports are sent.</li> </ul>
CLI admin password	The default value for CLI admin is <code>admin</code> . To change the password, you need to know the current password.  <p><b>Note:</b> Do not change the <code>current_admin_password</code>. The automated deployment uses the current password to reset the new password.</p>

**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
```

```

# 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

```

**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 abs-4.0.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 (`abs-4.0.tar.gz`) in `software` directory by entering the following command:

```
# tar -czf abs-4.0.tar.gz pingidentity/abs
```

### Change Dashboard default settings

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

Variable	Description
<code>enable_ui</code>	Configures whether the deployment package installs Dashboard UI or not. The default value is <code>true</code> .
	 <b>Note:</b> Dashboard and Syslog cannot be disabled together.

<code>enable_xpack</code>	Configures whether the deployment package installs X-pack. The default value is <code>true</code> .
<code>kibana_port</code>	Port used to access the PingIntelligence Dashboard.
<code>elastic_password</code>	Elasticsearch password. The default value is <code>changeme</code> .  <i>i</i> <b>Note:</b> Do not change the <code>elastic_password</code> after PingIntelligence installation is complete.
<code>kibana_password</code>	Kibana password. The default value is <code>changeme</code> .  <i>i</i> <b>Note:</b> Do not change the <code>kibana_password</code> after PingIntelligence installation is complete.
<code>ping_user_password</code>	Password for the default user name <code>ping_user</code> .
<code>ping_admin_password</code>	Password for the admin.
<b>Syslog configuration:</b> <ul style="list-style-type: none"> <li>▪ <code>enable_syslog</code></li> <li>▪ <code>host, port</code></li> <li>▪ <code>facility</code></li> </ul>	Configure Syslog details.  Setting <code>enable_syslog</code> to <code>true</code> lets Dashboard log detected attacks in syslog server.  <i>i</i> <b>Note:</b> Dashboard and Syslog cannot be disabled together.  Provide the host and port number of syslog server.
<code>restricted_user_access</code>	Defines the Dashboard user. Set it to <code>true</code> 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 <a href="#">ABS users for API reports</a> on page 170
CLI admin password	The default value for CLI admin is <code>admin</code> . To change the password, you need to know the current password.  <i>i</i> <b>Note:</b> Do not change the <code>current_admin_password</code> . The automated deployment uses the current password to reset the new password.

*i* **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:

```
---
dashboard:
```

```

  ui:
    # Configuration for Kibana UI. Set it to true to instal kibana and
    elasticsearch
    enable_ui: true

    # Install elasticsearch with xpack-enabled
    enable_xpack: true

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

  users
    # Passwords for elasticsearch, kibana, ping_user, and ping_admin
    # Dashboard will be accessible by these accounts
    # Please set strong passwords
    elastic_password: changeme
    kibana_password: changeme
    ping_user_password: changeme
    ping_admin_password: changeme

  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: true

  # CLI admin password
  current_admin_password: admin
  new_admin_password: admin

```

### Change AAD default settings

You can change the default settings in AAD by editing the `aad-defaults.yml` file. The following table lists the variables that you can set for AAD:

Variable	Description
<code>env</code>	<p>Set the key to <code>dev</code> if you are fetching API definitions in an API development environment.</p> <p>Set the key to <code>prod</code> if you are fetching API definitions in a production environment</p>
<code>fullsync</code>	<p>Set it to <code>true</code> if you want to reflect the APIs at source in ASE exactly. AAD deletes any APIs from ASE that is present in ASE but not present at the source.</p> <p>Set it to <code>false</code> if you want to synchronize the source APIs in ASE without deleting other APIs present in ASE.</p> <p><b>Note:</b> <code>fullsync</code> key is only valid in a development environment</p>



mode	<p>Set the value to <code>discovery</code> when ASE is deployed in inline mode.</p> <p>Set the value to <code>gateway</code> when ASE is deployed in sideband mode. For more information on AAD in <code>gateway</code> mode, see the Deployment Guide.</p> <p>Set the value to <code>pingaccess</code> when ASE is deployed in <code>sideband</code> mode with PingAccess. For more information on AAD in <code>pingaccess</code> mode, see the <a href="#">PingIntelligence for APIs – PingAccess Integration</a> guide.</p> <p>For more information on ASE modes, see the <a href="#">ASE Admin Guide</a>.</p>
<p><b>Gateway:</b></p> <ul style="list-style-type: none"> <li>▪ <code>management_url</code></li> <li>▪ <code>management_username</code></li> <li>▪ <code>management_password</code></li> </ul>	Provide the URL, management username and password of the API Gateway.
<p><b>PingAccess:</b></p> <ul style="list-style-type: none"> <li>▪ <code>management_url</code></li> <li>▪ <code>management_username</code></li> <li>▪ <code>management_password</code></li> </ul>	Provide the URL, management username and password of PingAccess. Enter the values for PingAccess when mode is set to <code>pingaccess</code>
CLI admin password	<p>The default value for CLI admin is <code>admin</code>. To change the password, you need to know the current password.</p> <p><b>Note:</b> Do not change the <code>current_admin_password</code>. The automated deployment uses the current password to reset the new password.</p>

**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 `aad-defaults.yml` file:

```

---
aad:
  env: prod
  fullsync: false

  # AAD mode. Valid values discovery, span_port, gateway, and pingaccess
  # discovery will pull discovered APIs from ABS
  # span_port will pull discovered APIs from ABS
  # gateway will pull APIs from Axway API Gateway
  # pingaccess will pull APIs from PingAccess
  mode: discovery

  gateway:
    # Configuration for gateway
    management_url: https://127.0.0.1:8075/
    management_username: apiadmin

```

```

management_password: changeme

pingaccess:
  # Configuration for gateway
  management_url: https://127.0.0.1:8075/
  management_username: Administrator
  management_password: changeme

# CLI admin password
current_admin_password: admin
new_admin_password: admin

```

## Step 2: 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](#).

```

[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: Fri Jun 07 06:05:25 EST 2019
Starting configure scripts
=====

Fri Jun 07 06:05:25 EST 2019: Setting up local environment
Fri Jun 07 06:05:25 EST 2019: Installing packages
Fri Jun 07 06:05:25 EST 2019: 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

Fri Jun 07 06:06:14 EST 2019: 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 3: 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: Fri Jun 07 06:06:22 EST 2019
Starting setup scripts
=====
Fri Jun 07 06:06:22 EST 2019: Setting up local environment
Fri Jun 07 06:06:22 EST 2019: Installing packages
Fri Jun 07 06:06:23 EST 2019: 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

Fri Jun 07 06:23:37 EST 2019: 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.

## 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`
  - **AAD:** `pi-aad.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
8. AAD

## Verify PingIntelligence Installation

Verify that all the components have installed and started successfully.

### Verify ASE installation

Log in to the ASE EC2 instance 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://<ip>:<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",
  "across_api_prediction_mode": true,
  "api_discovery": {
    "subpath_length": "1",
    "status": true
  },
  "apis": [
    {
      "api_name": "apiKeypubatmapp",
      "host_name": "*",
      "url": "/",
      "api_type": "regular",
      "creation_date": "Thu Oct 10 10:31:01 UTC 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": ""
```

```

        "token": true,
        "training_started_at": "n/a",
        "training_duration": "n/a",
        "prediction_mode": false,
        "apikey_header": "",
        "apikey_qs": "QS_API_KEY"
    },
    {
        "api_name": "ws",
        "host_name": "*",
        "url": "/app",
        "api_type": "decoy-incontext",
        "creation_date": "Thu Oct 10 10:31:01 UTC 2019",
        "servers": 1,
        "protocol": "ws",
        "cookie": "JSESSIONID",
        "token": false,
        "training_started_at": "Thu Oct 10 10:32:53 UTC 2019",
        "training_duration": "1 hour",
        "prediction_mode": true,
        "apikey_header": "",
        "apikey_qs": ""
    }
],
"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"
        }
    ]
},
"ase_logs": [
    {
        "ase_node": "317eale4-4f52-4784-9c3a-80f659ac6162",
        "last_connected": "Thu Oct 10 15:11:05 UTC 2019",
        "logs": {
            "start_time": "Thu Oct 10 10:31:06 UTC 2019",
            "end_time": "Thu Oct 10 15:11:05 UTC 2019",
            "gzip_size": "80.11MB"
        }
    }
],
"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
        }
    ]
}
}

```

### Verify Dashboard Installation

To verify the Dashboard installation, enter the kibana IP address from the hosts file in your web browser. Log in using username `ping_user` and the default password `changeme`.

See the ASE, ABS and Dashboard admin guides to 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:

- [PingIntelligence Apigee Integration](#) on page 460
- [PingIntelligence AWS API Gateway Integration](#) on page 476
- [Azure APIM sideband integration](#) on page 525
- [Axway sideband integration](#) on page 498

- [Mulesoft sideband integration](#) on page 561
- [NGINX sideband integration](#) on page 580
- [PingAccess sideband integration](#) on page 548
- [PingIntelligence WSO2 integration](#) on page 599

**Inline configuration:** If you configured PingIntelligence ASE as [Inline ASE](#) on page 75, 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 PingIntelligence for APIs Dashboard](#): Access the main PingIntelligence for APIs Dashboard with a browser at this URL: `https://<kibana ip>:<kibana port>/`. In the above URL, Kibana IP is the IP address of the Kibana VM configured in `config/hosts` file.

Login to PingIntelligence for APIs 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 406. 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: Fri Jun 07 07:23:11 EST 2019
Starting stop scripts
=====
Fri Jun 07 07:23:11 EST 2019: 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
Fri Jun 07 07:32:53 EST 2019: Stop successful
=====
```

## Logs

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

## 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 utilizes Elasticsearch and Kibana to provide a graphical view of an API environment including traffic metrics, attack types and blocked connections. The PingIntelligence for APIs Dashboard makes periodic REST API calls to an ABS engine which returns JSON reports that are used to generate graphs. Organizations can utilize the PingIntelligence for APIs Dashboard examples to develop direct integration into in-house graphical management systems.

#### Users

You can install all the PingIntelligence products either as a root user or a non-root user. Make sure that the entire deployment is a homogenous deployment. Either all the products should be installed as a root user or as a non-root 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 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.

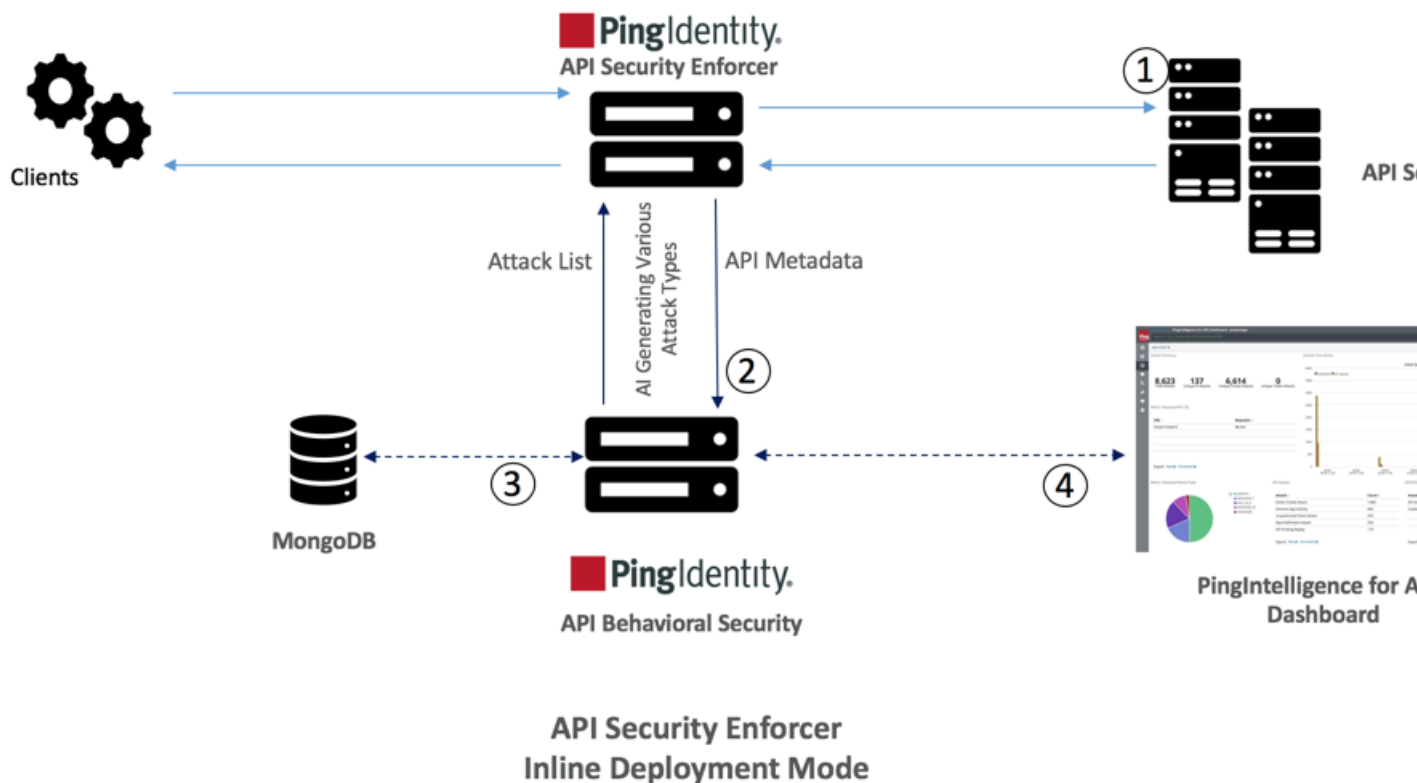
**i Important:** The management ports 8010 and 8020 should not be exposed to the Internet. If you are setting up the deployment in an AWS environment with security groups, use private IPs for ASE to ABS connections to avoid security group issues.

## 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.



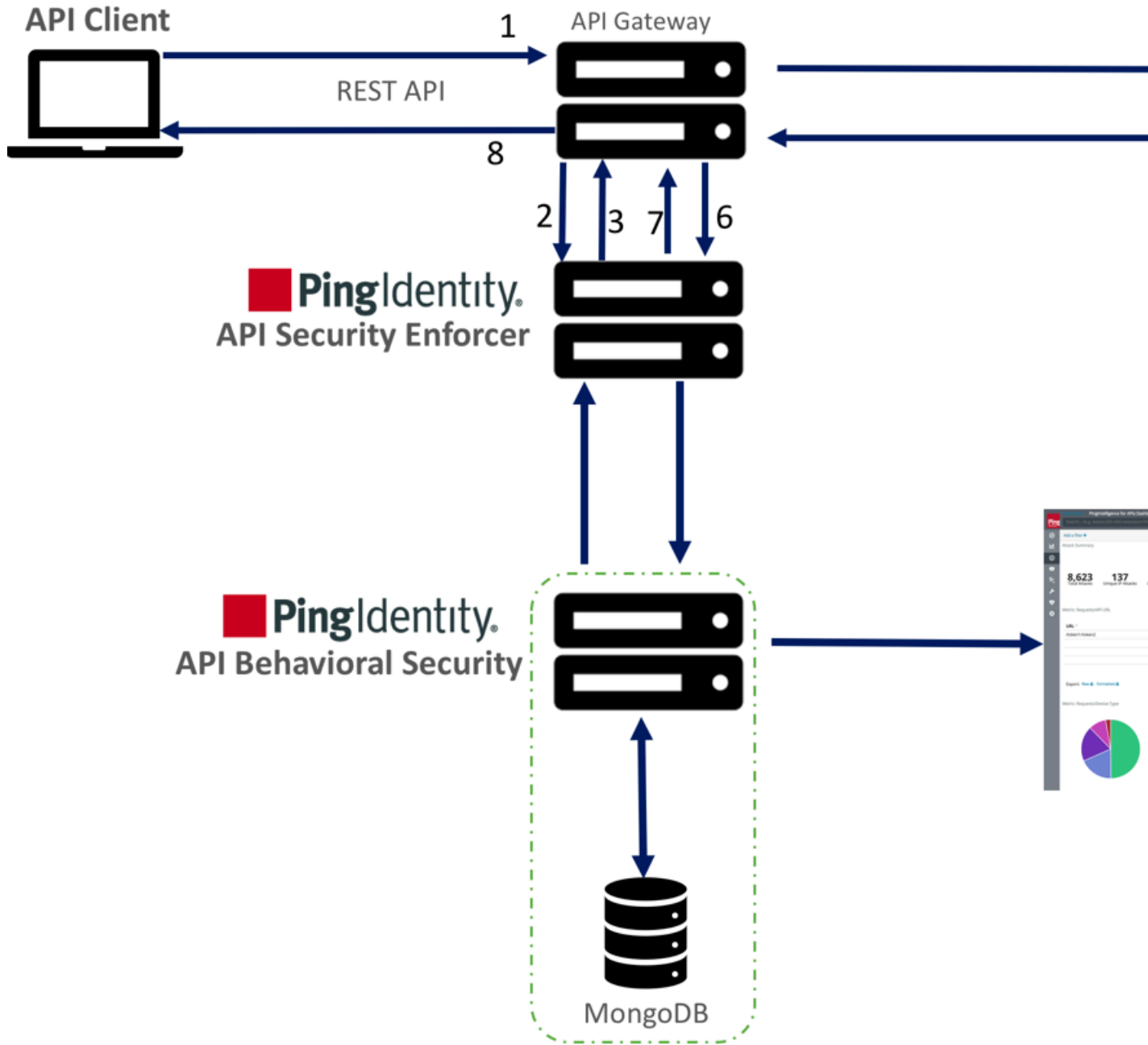
To configure ASE to work in the Inline mode, set the `mode=inline` in the `/opt/pingidentity/ase/config/ase.conf` file.

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 ase-rhel-4.0.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 ase-rhel-4.0.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.

### 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

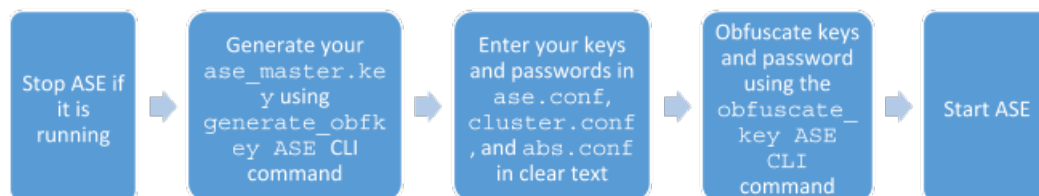
You must obfuscate the keys and passwords configured in `ase.conf`, `cluster.conf`, and `abs.conf` in the config directory. ASE ships with a default `ase_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `ase_master.key`.

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.

**i 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.

**i 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

**i 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.

**i 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.

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.

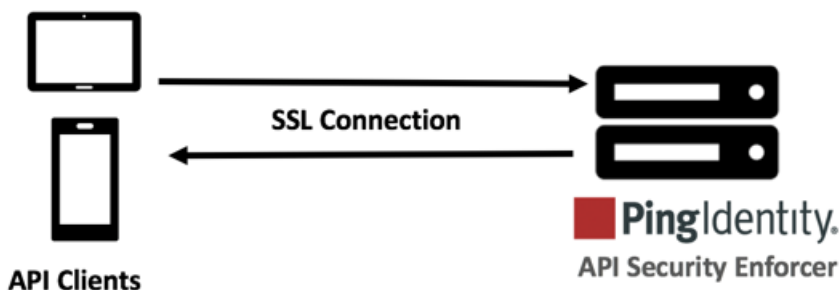
```
/opt/pingidentity/ase/bin/stop.sh -u admin -p admin
checking API Security Enforcer status...
sending stop request to ASE. please wait...
API Security Enforcer stopped
```

## 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.

**i 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.

**i 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 B – 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 abs-4.0.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.

## 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

The transaction count is updated every day after 00:00 hours midnight in the `/opt/pingidentity/abs/logs/abs.log` file. To check the current monthly transaction count, `grep` for `Current Transactions` in the `abs.log` file. Following is a sample snippet for the current number of transactions:

```
$ grep "Current Transactions" *
abs.log:2019-05-19 00:01:25 INFO - Current Transactions: 289088158 between
earlier date: Wed May 01 00:00:00 EST 2019 and later date: Sat May 18
23:59:59 EST 2019
```

The `earlier date` is always the starting date of the month.

## Obfuscate passwords

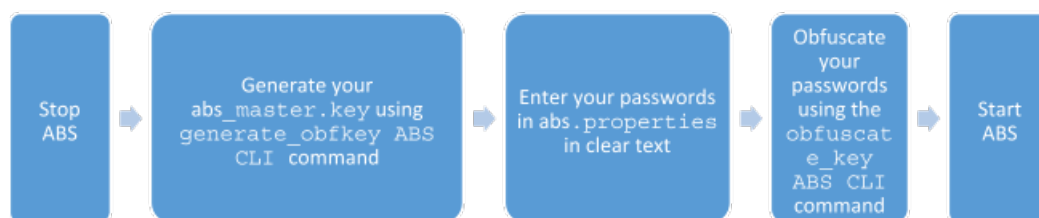
Using ABS command line interface, you can obfuscate the keys and passwords configured in `abs.properties`. The following keys and passwords are obfuscated:

- `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.

**i 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.

**i 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 431 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_certificate.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>.

**i 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.

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 4.0.6 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.

**i 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:

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:

```
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 4.0.6
connecting to: admin
switched to db abs_metadata
WriteResult({ "nInserted" : 1})
bye
```

## Start and Stop ABS

### Prerequisite:

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 executing the start script.

### 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:

```
$ /opt/pingidentity/abs/bin/start.sh
Starting API Behavioral Security 4.0...
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

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
```

## 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_ase.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 `abs_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:~N6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0

; secret key for abs node
```

```
secret_key=OBF:AES:Y2DadCU4JFZp3bx8Ehn0iw: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 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](#)
- [Define and add an API JSON](#)

For more information on ASE sideband deployment, see [Sideband API Security Enforcer](#).

### 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 the *ABS Admin Guide*.

```
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"
});
```

### 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 Jun 06 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 software

The PingIntelligence for APIs Dashboard installation process is summarized below:

- Install PingIntelligence for APIs Dashboard
- Obfuscate keys and password
- Install Elasticsearch
- Install Kibana
- Integrate Dashboard with ABS AI engine
- Start PingIntelligence for APIs Dashboard

### Install PingIntelligence for APIs dashboard

#### Prerequisites

1. `wget` and `openssl` must be installed on your system
2. PingIntelligence for APIs Dashboard, Elasticsearch and Kibana should run as a non-root user

#### Install PingIntelligence for APIs Dashboard

Download PingIntelligence for APIs Dashboard from the [download](#) site to a Linux server. PingIntelligence Dashboard engine also provides support for Syslog. Dashboard only sends the attack data to Syslog. Enabling or disabling syslog server logging is configured in `dashboard.properties` file. Complete the following steps to install PingIntelligence Dashboard:

1. Start `shell` as a non-root user

**2. Change the directory to /opt**

```
$cd /opt
```

**3. Create a pingidentity directory**

```
$ sudo mkdir pingidentity
```

**4. Change the permissions for the pingidentity directory. The pingidentity directory will be owned by a non-root user.**

```
$ sudo chown -R "$(id -nu):$(id -ng)" /opt/pingidentity
```

**5. Install PingIntelligence for APIs Dashboard**

```
$ tar -zxf dashboard-4.0.tar.gz
```

The following table shows the directories created when PingIntelligence for APIs Dashboard is installed:

Directories	Description
bin	ABS Start and Stop scripts; Elasticsearch and Kibana initialization scripts.
config	dashboard.properties file used to configure PingIntelligence for APIs Dashboard A subdirectory called dashboard containing Kibana schema for each API
data	Temporary storage for ABS data
lib	Contains dashboard.jar and dependent external jar files
logs	Contains PingIntelligence for APIs Dashboard log files which are rotated every 24 hours
plugin	Contains Dashboard UI plug-in. Do not edit the contents of the directory.
util	Contains the check_ports_dashboard.sh script to check the availability of default Elasticsearch and ABS ports to connect.

**Change Dashboard default settings**

It is recommended that you change the default settings in Dashboard. Complete the following steps to change the default settings:

**Change dashboard\_master.key**

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

**Command:** generate\_obfkey.

**Change abs\_master.key:** Enter the generate\_obfkey command to change the default ABS master key:

```
/opt/pingidentity/dashboard/bin/cli.sh generate_obfkey -u admin -p
Password>
```

Please take a backup of config/dashboard\_master.key before proceeding.

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

```
Warning: Obfuscation master key file /opt/pingidentity/dashboard/config/
dashboard_master.key already exist. This command will delete it create a new
key in the same file
```

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

```
creating new obfuscation master key
Success: created new obfuscation master key at /opt/pingidentity/dashboard/
config/dashboard_master.key
```

### Change CLI admin password

Dashboard ships with the default user `admin` and the default password `admin`. You can change the default password by using the `update_password` Dashboard CLI command:

```
/opt/pingidentity/dashboard/bin/cli.sh -u admin update_password -p
Password>

New Password>
Re-enter New Password>
Success. Password updated for CLI
```

### Change default passwords

Navigate to `config` directory and edit the `dashboard.properties` file to change the following values:

- `dashboard.properties` file – `abs.access_key`, `abs.secret_key`, `es.password`

### Repackage Dashboard

Tar Dashboard after changing the default values. Enter the following command:

```
tar -zcvf dashboard-4.0 pingidentity/
```

Make sure that the original file name is retained when you tar Dashboard and the file is saved in the `software` directory on the management instance.

### Obfuscate keys and passwords

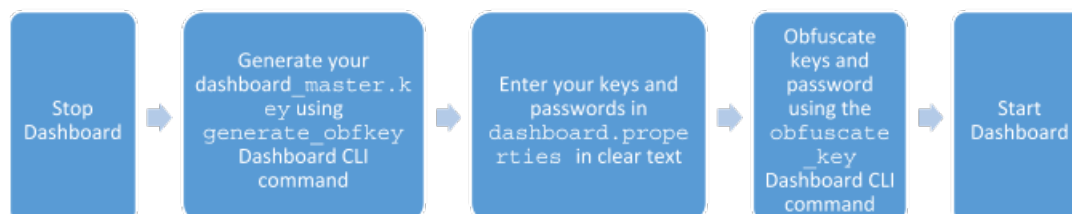
Using Dashboard's command line interface, you can obfuscate the keys and passwords configured in `dashboard.properties`. The following keys and passwords are obfuscated:

- `abs.access_key`
- `abs.secret_key`
- `es.password`

Dashboard ships with a default `dashboard_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `dashboard_master.key`.

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

The following diagram summarizes the obfuscation process:



### Generate `dashboard_master.key`

You can generate the `dashboard_master.key` by running the **`generate_obfkey`** command in the Dashboard CLI:

```

/opt/pingidentity/dashboard/bin/cli.sh generate_obfkey -u admin -p
Password>

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

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

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

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

creating new obfuscation master key
Success: created new obfuscation master key at /opt/pingidentity/dashboard/
config/dashboard_master.key

```

### Obfuscate key and passwords

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

```

/opt/pingidentity/dashboard/bin/cli.sh obfuscate_keys -u admin -p
Password>

Please take a backup of config/dashboard.properties before proceeding

Enter clear text keys and password before obfuscation.

Following keys will be obfuscated
config/dashboard.properties: abs.access_key, abs.secret_key and es.password

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

obfuscating /opt/pingidentity/dashboard/config/dashboard.properties

Success: secret keys in /opt/pingidentity/dashboard/config/
dashboard.properties obfuscated

```

**i Important:** After the keys and passwords are obfuscated and the Dashboard has started, move the `dashboard_master.key` to a secure location away from the Dashboard.

### Install Elasticsearch

Complete the following steps to download and install Elasticsearch:

1. Start `shell` as a non-root user
2. Change the directory to `/opt`

```
$cd /opt
```

3. Create an elasticsearch directory

```
$ sudo mkdir elasticsearch
```

4. Change the permissions for the `elasticsearch` directory. The `elasticsearch` directory will be owned by a non-root user.

```
$ sudo chown -R "$(id -nu):$(id -ng)" /opt/elasticsearch
```

5. Change directory to `elasticsearch`

```
$ cd /opt/elasticsearch
```

6. Download Elasticsearch:

```
$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.8.1.tar.gz
```

7. Install Elasticsearch:

```
$ tar -zxvf elasticsearch-6.8.1.tar.gz
```

8. Change directory:

```
$ cd /opt/elasticsearch/elasticsearch-6.8.1
```

**Change heap size of Elasticsearch for production deployments:** Complete the following steps:

1. Navigate to `/config/jvm.options` to edit the `jvm.options` file
2. Find the following section in the `jvm.options` file:

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms1g
-Xmx1g
```

3. Change the value of `-Xms1g` to `-Xms4g` and the value of `-Xmx1g` to `-Xmx4g`
4. Save the file

### Configure Elasticsearch

Configure Elasticsearch by running the `dashboard_elasticsearch_init.sh` script located in the `Dashboard bin` directory. The `dashboard_elasticsearch_init.sh` script asks for the full path where you have saved the CA signed certificate. If you do not have a CA signed certificate, generate a self-signed certificate without a passphrase using the OpenSSL commands.

```
$ /opt/pingidentity/dashboard/bin/dashboard_elasticsearch_init.sh
```

```
[pingidentity@localhost ~]$ /opt/pingidentity/dashboard/bin/
dashboard_elasticsearch_init.sh
updating elasticsearch configuration

Enter SSL CA Signed Certificate path >(full path)
Enter SSL Private Key Path >(full path)

enter pkcs#12 keystore new password >
enter pkcs#12 keystore new password again >

creating elasticsearch config keystore
config keystore created

creating password protected pkcs#12 keystore for private key and certificate
pkcs#12 keystore created at config/ssl/elastic-certificates.p12
```

```
Starting Elasticsearch to update default passwords. Please wait for 15
seconds.
Elasticsearch started with pid 2532 and listening at https://localhost:9200

updating default user passwords

## elastic [superuser] password. Remember this password for the Dashboard
setup
enter elastic user new password >
enter elastic user password again >
password updated for user elastic

## kibana [kibana user] password. Remember this password for the Kibana
setup
enter kibana user new password >
enter kibana user password again >
password updated for user kibana

Elasticsearch configuration is complete. Elasticsearch is running at
https://localhost:9200
[pingidentity@localhost ~]$
```

## Install Kibana

Complete the following steps to install Kibana:

1. Start shell as a non-root user
2. Change the directory to /opt

```
$cd /opt
```

3. Create a kibana directory

```
$ sudo mkdir kibana
```

4. Change the permissions for the kibana directory to ownership by a non-root user.

```
$ sudo chown -R "$(id -nu):$(id -ng)" /opt/kibana
```

5. Change directory to kibana

```
$ cd /opt/kibana
```

6. Download Kibana: \$ wget "[https://artifacts.elastic.co/downloads/kibana/kibana-6.8.1-linux-x86\\_64.tar.gz](https://artifacts.elastic.co/downloads/kibana/kibana-6.8.1-linux-x86_64.tar.gz)"

7. Install Kibana:

```
$ tar -zxf kibana-6.8.1-linux-x86_64.tar.gz
```

8. Change directory:

```
$ cd /opt/kibana/kibana-6.8.1-linux-x86_64
```

### Note:

By default, the Kibana uses port 443 with su/sudo access. If you want to use any other port, for example 5601, use:

```
$ export KIBANA_DEFAULT_PORT=5601
```



If you are a non-root user, use ports greater 1024.

**Initialize Kibana:** After installing Kibana, initialize Kibana by running the following command:

```
$ /opt/pingidentity/dashboard/bin/dashboard_kibana_init.sh
```

```
[pingidentity@localhost ~]$ /opt/pingidentity/dashboard/bin/
dashboard_kibana_init.sh
updating Kibana configuration
Enter SSL CA Signed Certificate path >(full path)
Enter SSL Private Key Path >(full path)
enter kibana [kibana user] password >
enter kibana [kibana user] password again >
Kibana configuration is complete.
Starting Kibana in the background...
Kibana started with pid 2535 and listening at https://[0.0.0.0]
```

### Install Ping styling plug-in for Kibana

Install the Ping styling plug-in for Kibana by entering the following command:

```
./bin/kibana-plugin install
file:///opt/pingidentity/dashboard/plugins/pingstyling-4.0.zip
```

### Integrate dashboard with ABS AI engine and syslog server

For production environments with high traffic loads, it is recommended to install one or more dedicated ABS nodes for PingIntelligence for APIs Dashboard processing. Install an ABS node (see [Install ABS AI engine software](#) on page 430) and set `dashboard_node` to `true` in the `abs.properties` file (`/opt/pingidentity/config/`). This ABS node will be used exclusively to process reports for the Dashboard; no access log processing occurs on this node.

To configure the Dashboard, edit the `dashboard.properties` file which is part of the `config` directory created when the Dashboard was installed. Set the Elasticsearch password to match the password used when configuring Elasticsearch.

```
# 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
+FKmL5cNaIDrfVuyc5hIYjimqh7Rnf3bv9hW0+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:
+ZGHbhpY4HENYYqJ4wn50AmoO6CZ30cfjqTYQCfGbgc=
# kibana version
kibana.version=6.6.0

### 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

```

Configure all parameters in the `dashboard.properties` file:

Parameter	Description
abs.host	IP address of the ABS server  <div style="border: 1px solid gray; padding: 5px;"> <p> <b>Note:</b> Two options exist to choose an ABS server: 1) Utilize an existing ABS server. 2) For production deployments, Ping Identity recommends dedicating an ABS node exclusively for the Dashboard.</p> </div>
abs.port	REST API port number of the ABS host – See <code>abs.properties</code> Default value is 8080
abs.ssl	Setting the value to true ensures SSL communication between ABS and PingIntelligence for APIs Dashboard
abs.restricted_user	When set to true, Elasticsearch uses the restricted user header (configured in <code>pingidentity/abs/mongo/abs_init.js</code> 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 <a href="#">ABS users for API reports</a>

abs.access_key	Access key from ABS – See <code>pingidentity/abs/mongo/abs_init.js</code> . Make sure to enter the access key based on the value set in the previous variable. For example, if <code>abs.restricted_user</code> is set to true, then enter the access key for restricted user. If <code>abs.restricted_user</code> is set to false, then use the access key for the admin user.
abs.secret_key	Secret key from ABS – See <code>pingidentity/abs/mongo/abs_init.js</code> . Make sure to enter the secret key based on the value set in the previous variable. For example, if <code>abs.restricted_user</code> is set to true, then enter the secret key for restricted user. If <code>abs.restricted_user</code> 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 value is 30 mins and default value is 60 mins.
es.url	Elasticsearch URL
es.username	Elasticsearch username
es.password	Elasticsearch password.
kibana.version	Kibana version - default is 6.4.3
dashboard.log.level	Log level for Dashboard Default log level is <code>INFO</code> . Another log level is <code>DEBUG</code>

### Start PingIntelligence for APIs Dashboard

To start the PingIntelligence for APIs Dashboard, navigate to the `/opt/pingidentity/dashboard/bin` directory and enter the following command:

```
[pingidentity@localhost bin]# ./start.sh
Dashboard 4.0 starting...
Please see /opt/pingidentity/dashboard/logs/dashboard.log for more details
```

After PingIntelligence for APIs Dashboard is started, wait 15 seconds for the Dashboard to create the following two users:

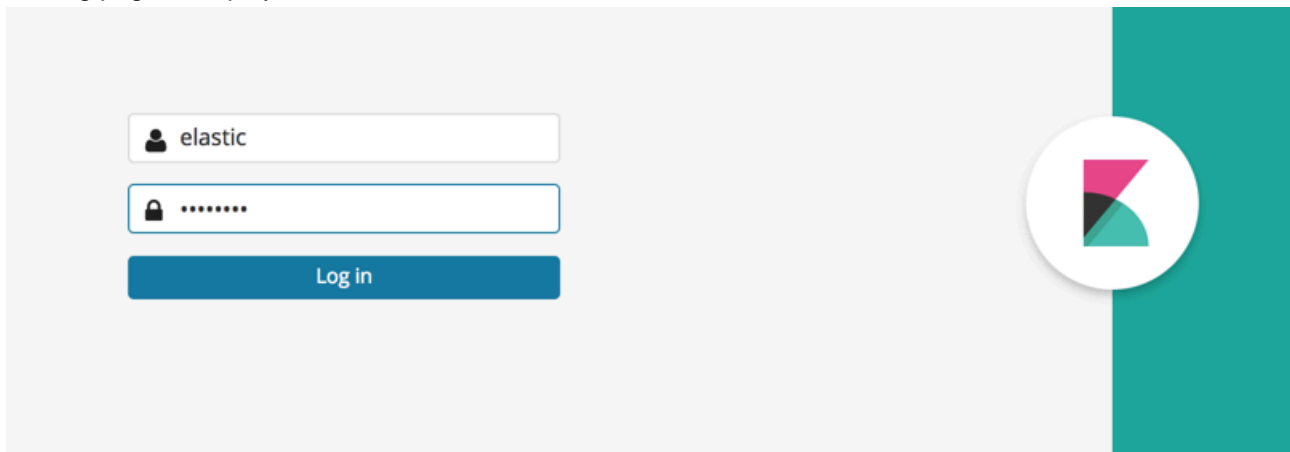
- `ping_admin`
- `ping_user`

**Note:** Immediately after starting PingIntelligence for APIs Dashboard, change the password for both the users.

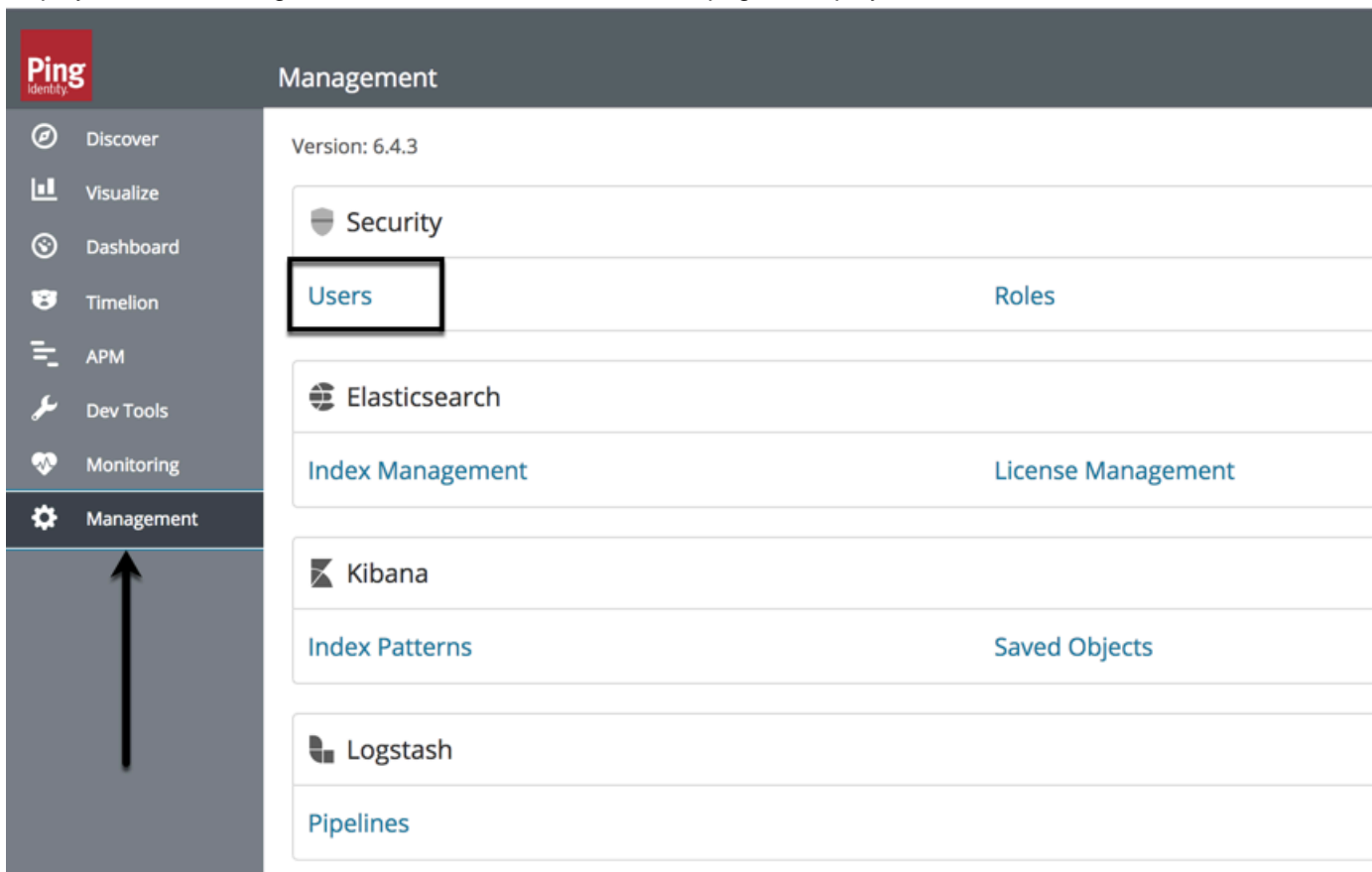
### Connect to the dashboard

Access <https://<ip:port>/app/kibana#/dashboard/pingapiintelligence> to load the main dashboard. In the above link, `<ip:port>` is the IP address and port (default – 443) configured in `kibana.yml`. Change the password of users `ping_admin` and `ping_user` by completing the following steps:

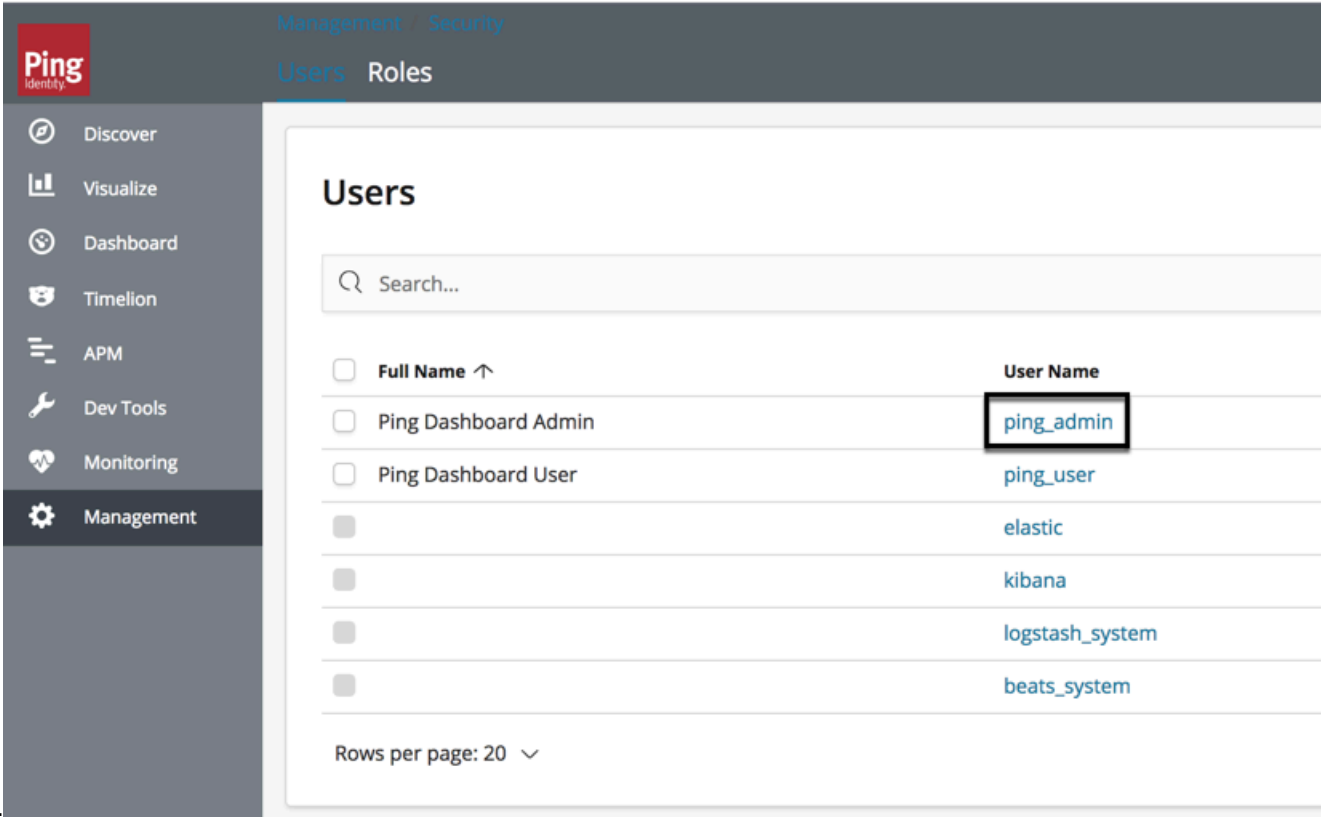
1. Log in using `elastic` user and the password set during [Elasticsearch configuration](#). The Kibana landing page is displayed.



2. In the Kibana landing page, click **Management**. The **Management** page is displayed. In the Management tab, click **Users**. The **Users** page is displayed:



3. On the **Users** page, click **ping\_admin** to change the email and password of **ping\_admin**



The screenshot shows the Ping Identity Users management interface. The left sidebar contains navigation options: Discover, Visualize, Dashboard, Timelion, APM, Dev Tools, Monitoring, and Management (highlighted). The main content area is titled 'Users' and features a search bar and a table of users. The table has columns for selection, Full Name, and User Name. The 'ping\_admin' user is highlighted with a black box.

<input type="checkbox"/>	Full Name ↑	User Name
<input type="checkbox"/>	Ping Dashboard Admin	ping_admin
<input type="checkbox"/>	Ping Dashboard User	ping_user
<input type="checkbox"/>		elastic
<input type="checkbox"/>		kibana
<input type="checkbox"/>		logstash_system
<input type="checkbox"/>		beats_system

Rows per page: 20 ▾

user.

4. On the **ping\_admin Users** page, update the **Email** and **Password** and click

**Edit "ping\_admin" user**

Username  
ping\_admin  
Username's cannot be changed after creation.

Full name  
Ping Dashboard Admin

**Email address**  
admin@index  
A valid email address is required

Roles  
ping\_admin\_role

**Password**  
.....

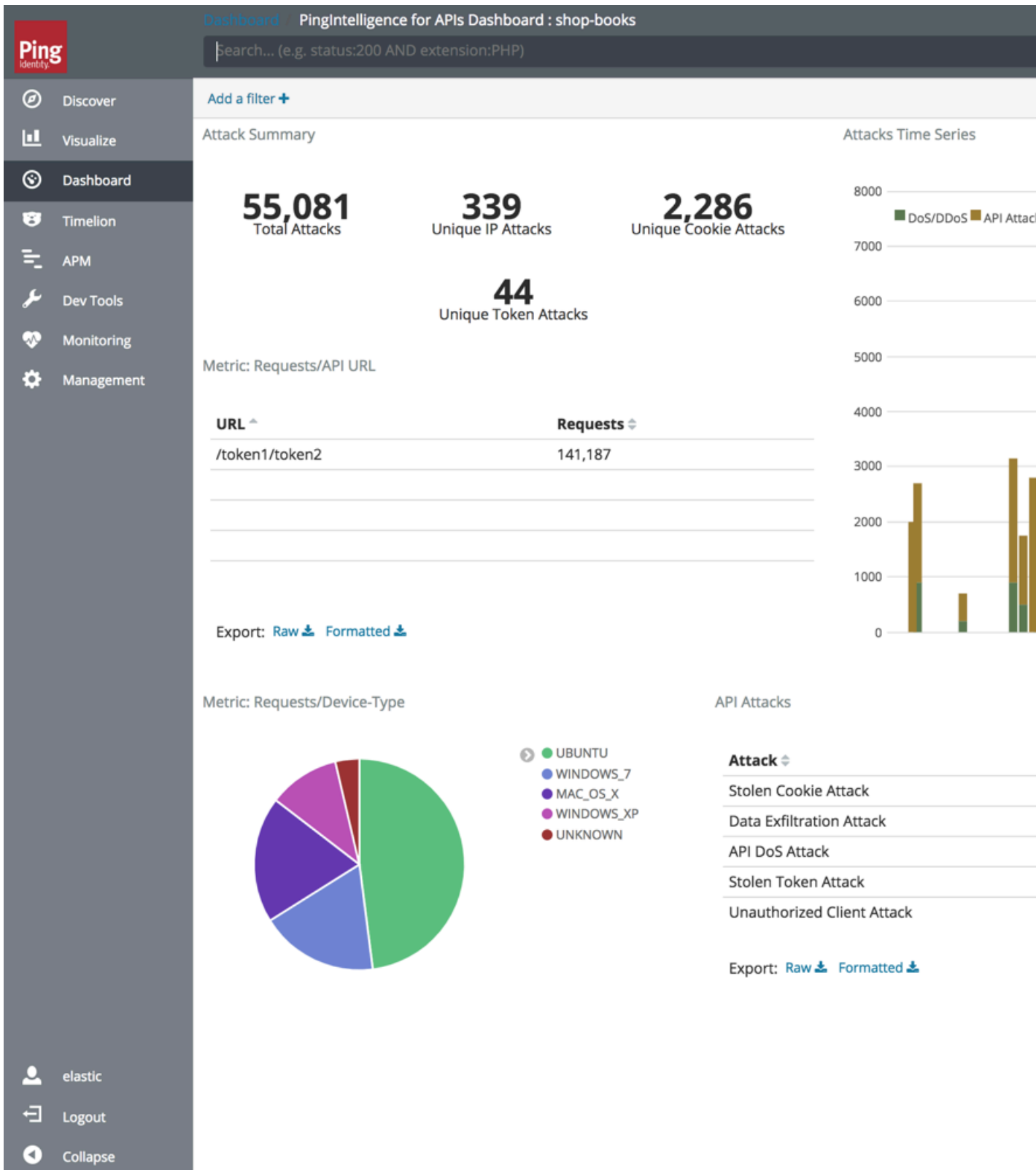
**Confirm password**  
.....

Save password Cancel

Update user Cancel Delete user

**Save:**

Repeat steps 2 through 4 for **ping\_user** to update **Email** and **Password**. Then log in with ping\_user credentials to view the dashboard. Here is a partial screen grab of the main dashboard:



### 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.

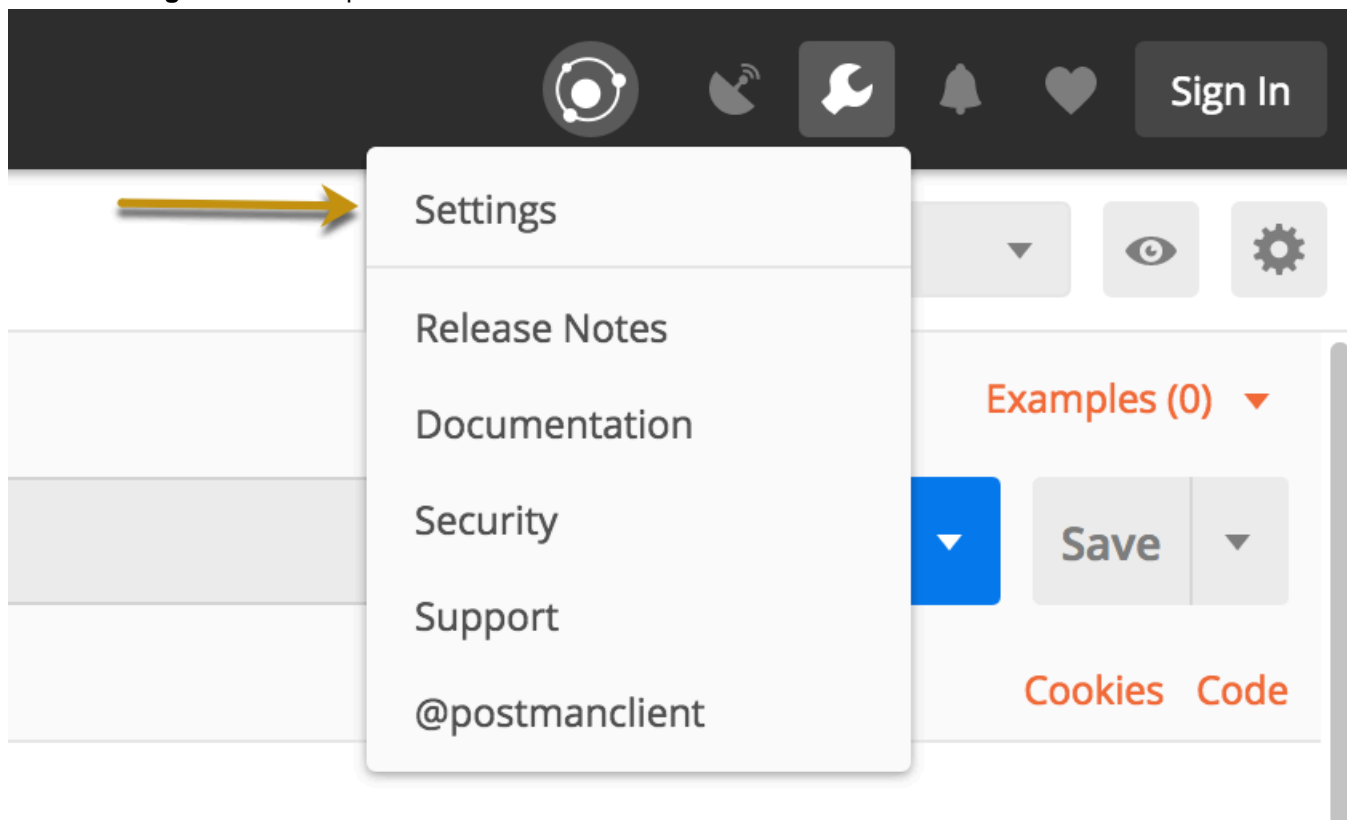
### Using ABS self-signed certificate with Postman

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.

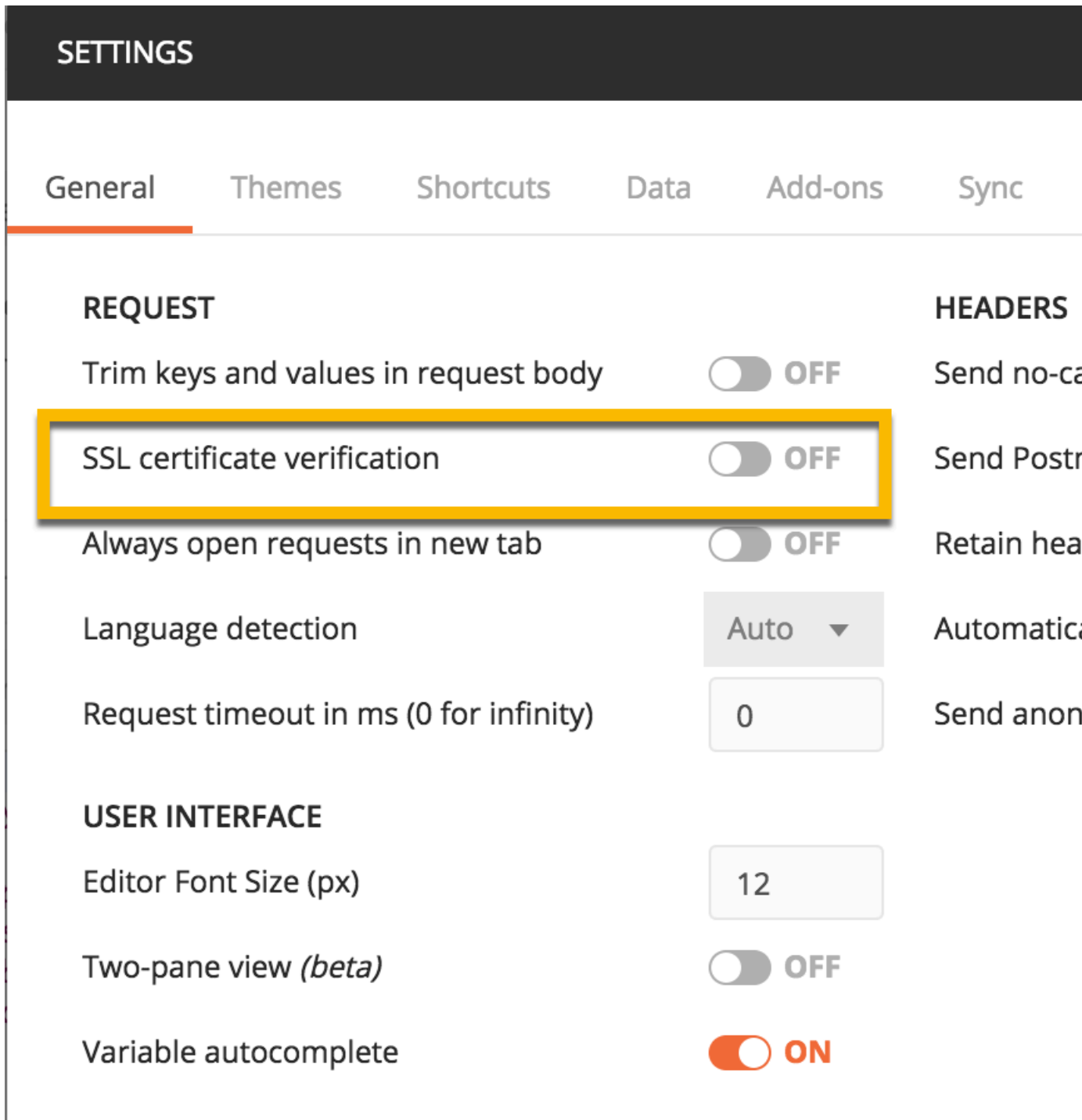
Click on the **spanner**  on the top-right corner of Postman client. A drop-down window is displayed.

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:



The screenshot shows the Postman Settings window with the 'General' tab selected. The 'REQUEST' section is visible, and the 'SSL certificate verification' toggle switch is highlighted with a yellow box and is currently turned off. Other settings in the 'REQUEST' section include 'Trim keys and values in request body' (OFF), 'Always open requests in new tab' (OFF), 'Language detection' (Auto), and 'Request timeout in ms (0 for infinity)' (0). The 'USER INTERFACE' section includes 'Editor Font Size (px)' (12), 'Two-pane view (beta)' (OFF), and 'Variable autocomplete' (ON).

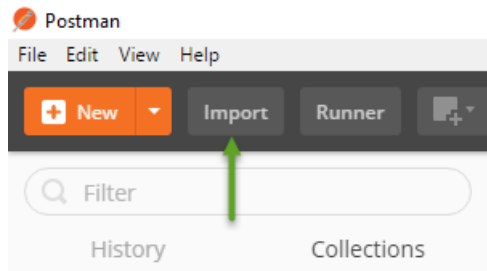
Setting	Value
Trim keys and values in request body	OFF
SSL certificate verification	OFF
Always open requests in new tab	OFF
Language detection	Auto
Request timeout in ms (0 for infinity)	0
Editor Font Size (px)	12
Two-pane view (beta)	OFF
Variable autocomplete	ON

### View ABS Reports in Postman

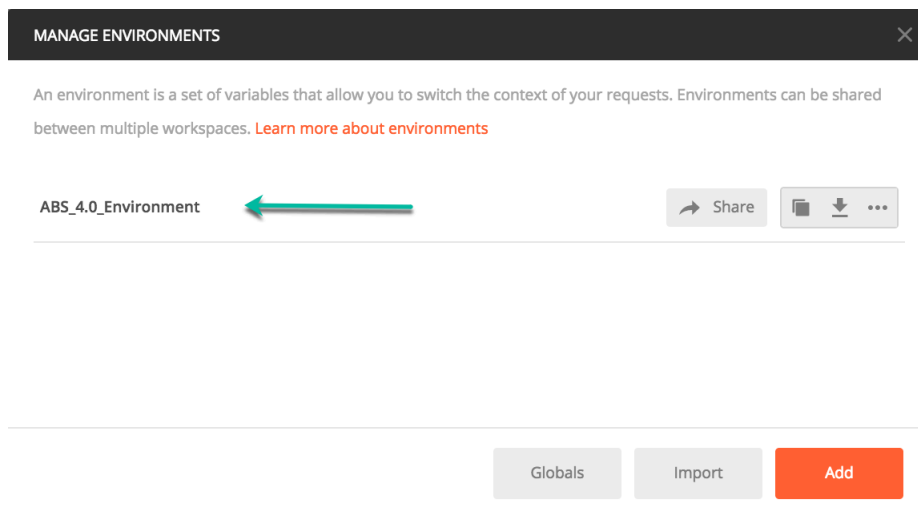
To view the reports, complete the following steps:

1. Download `ABS_4.0_Environment` and `ABS_4.0_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\_4.0\_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`.

MANAGE ENVIRONMENTS
✕

Environment Name

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	***	Persist All	Reset All
<input checked="" type="checkbox"/>	Server	192.168.11.166	192.168.11.166			
<input checked="" type="checkbox"/>	Port	8080	8080			
<input checked="" type="checkbox"/>	Access_Key_Header					
<input checked="" type="checkbox"/>	Secret_Key_Header					
<input checked="" type="checkbox"/>	Access_Key					
<input checked="" type="checkbox"/>	Secret_key					
<input checked="" type="checkbox"/>	API_Name	atmapp1	atmapp1			
<input checked="" type="checkbox"/>	Later_Date	2018-12-31T18:00	2018-12-31T18:00			
<input checked="" type="checkbox"/>	Earlier_Date	2018-10-25T13:30	2018-10-25T13:30			
<input checked="" type="checkbox"/>	System_URL	https://{{Server}}:{{P...	https://{{Server}}:{{Port}}/v3/abs			
<input checked="" type="checkbox"/>	System_Admin	{{System_URL}}/admin	{{System_URL}}/admin			
<input checked="" type="checkbox"/>	System_Metrics	{{System_URL}}/metr...	{{System_URL}}/metrics?later_date={{Later_Date}}&earl...			
<input checked="" type="checkbox"/>	System_API_Key_Met...	{{System_URL}}/apik...	{{System_URL}}/apikeys?later_date={{Later_Date}}&earl...			
<input checked="" type="checkbox"/>	System_OAuth_Toke...	{{System_URL}}/oaut...	{{System_URL}}/oauthtokens?later_date={{Later_Date}}...			

ⓘ Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. [Change](#) this behaviour from Settings. [Learn more about variable values](#)

Cancel
Update

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:

- [PingIntelligence Apigee Integration](#) on page 460
- [PingIntelligence AWS API Gateway Integration](#) on page 476
- [Azure APIM sideband integration](#) on page 525
- [Axway sideband integration](#) on page 498
- [Mulesoft sideband integration](#) on page 561
- [NGINX sideband integration](#) on page 580
- [PingAccess sideband integration](#) on page 548
- [PingIntelligence WSO2 integration](#) on page 599

## API Gateway Integration

---

### Apigee API gateway integration

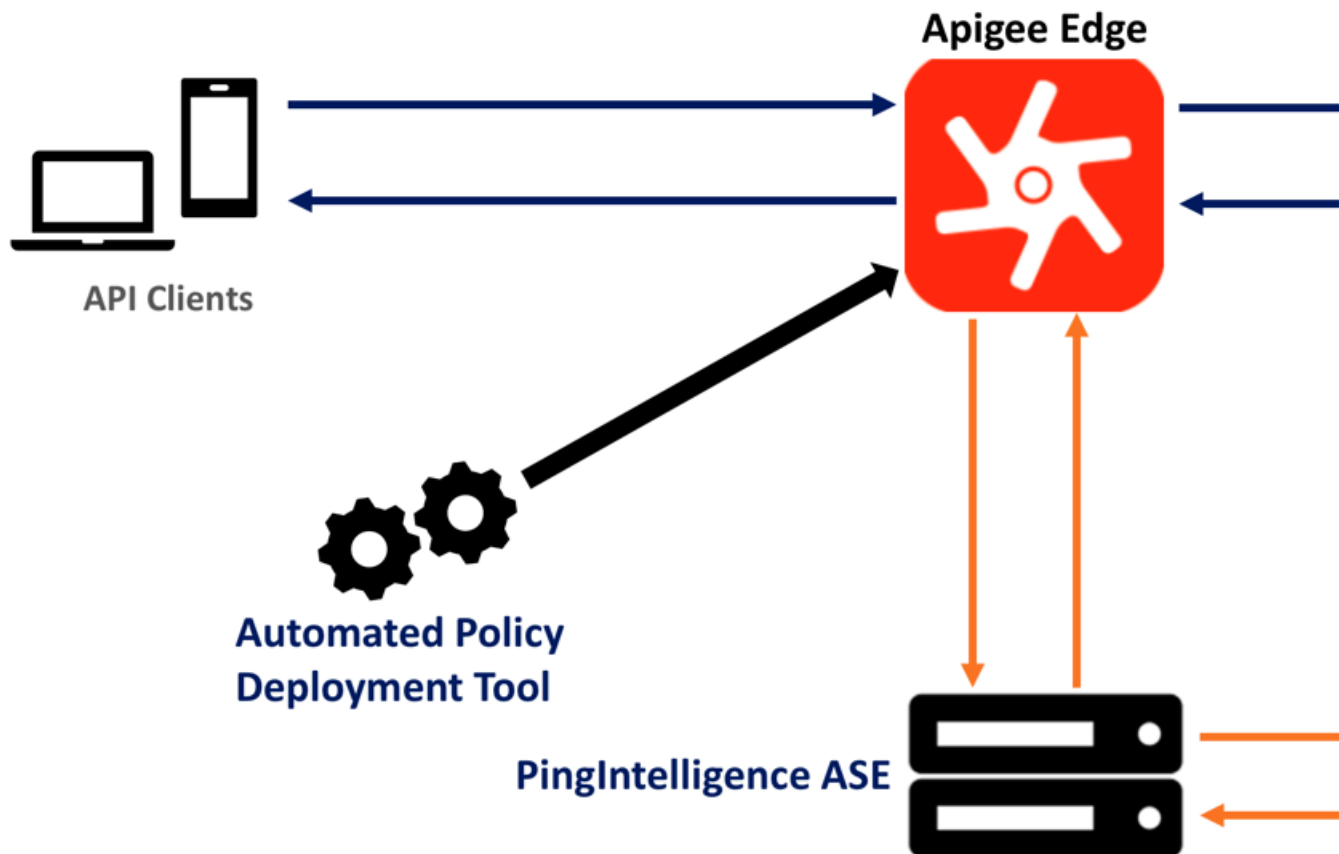
---

#### 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 policies.

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 works with Apigee Edge Cloud 18.12.04
- Machine where PingIntelligence Apigee tool is installed has OpenJDK 11 installed.

- **PingIntelligence software installation**

PingIntelligence 4.0 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 and 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-apigee-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: <ul style="list-style-type: none"> <li>deploy.sh: The script to deploy the PingIntelligence policy.</li> <li>undeploy.sh: The script to undeploy the PingIntelligence policy.</li> <li>status.sh: Reports the deployment status and configured Apigee credentials.</li> </ul>
lib	Jar files and various dependencies. Do not edit the contents of this directory.
policy	Contains the shared flows: <ul style="list-style-type: none"> <li>request_sharedflow.zip: Shared flow policy for request</li> <li>response_sharedflow.zip: Shared flow for response.</li> </ul> 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.

#### Configure the automated tool

Configure the apigee.properties file available in the /pingidentity/pi/apigee/config/ directory. The following table describes the various variables of the apigee.properties 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.
apigee_url	URL to connect to Apigee Edge
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	<p>The ASE secondary host IP address and port or hostname and port.</p> <p><b>Note:</b> This field cannot be left empty. In a testing environment, you can provide the same IP address for primary and secondary ASE host.</p>
ase_ssl	<p>Enable or disable SSL communication between Apigee Edge and ASE. The default value is <code>true</code>.</p>
ase_sideband_token	<p>Configure the ASE token generated during the <a href="#">prerequisite</a> step.</p>

Following is a sample `apigee.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.

#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
```

## 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



## Specs

Describe your services using the OpenAPI Specification format



## API Proxies

Route, transform, and secure your traffic through the API gateway



## Portals

Publish APIs and reference docs, and on-board developers



## Learn More

Tutorials, tips, and documentation for the new Edge experience

Proxy.

2. Click on the API name on which you want to apply the policy. The Develop page is displayed:

**Proxies**

Environment: prod | All

NAME	STATUS	TR
shop	●	0
...	●	0
...	●	0

3. On the Develop page, click on the **DEVELOP** tab:

API Proxies > shop > Develop > 1

Project Save Revision 1 Tools Deployment

Navigator << Flow: PreFlow

- shop
  - Policies
    - PI Request Flow Callout
    - PI Response Flow Callout
    - Response Flow Callout
  - Proxy Endpoints
    - default
      - All PreFlow
      - All PostFlow
  - Target Endpoints
    - default
      - All PreFlow
      - All PostFlow
  - Scripts

+ Step

4. In the **DEVELOP** tab, choose **PreFlow** under **Proxy Endpoints**, and click on **+ Step** for request. The Add Step window is displayed:

API Proxies > shop > Develop > 1

Project Save Revision 1 Tools Deployment

Navigator << Flow: PreFlow

- shop
  - Policies
    - PI Request Flow Callout
    - PI Response Flow Callout
    - Response Flow Callout
  - Proxy Endpoints
    - default
      - All PreFlow
      - All PostFlow
  - Target Endpoints
    - default
      - All PreFlow
      - All PostFlow
  - Scripts

+ Step

5. In the Add Step window, select **Flow Callout**. From the **Shared Flow** drop down list, select the Request rule and click on **Add**:

Add Step

Policy Instance

XSL Transform

SOAP Message Validation

Assign Message

Extract Variables

Access Entity

Key Value Map Operations

EXTENSION

Java Callout

Python

JavaScript

Service Callout

**Flow Callout**

Statistics Collector

Message Logging

Policy Type  Flow Callout

Display Name

Name

Shared Flow

Flow Callouts let

PingIntelligence-Request-Shared-Rule-No-Unencrypted-KVM

PingIntelligence-Response-Shared-Rule-No-Unencrypted-KVM

6. Repeat step 5 for Response rule.

7. Request and Response rules are added. Click on Save:

The screenshot shows the API Gateway console interface. On the left, a sidebar contains navigation options: 'DEVELOP' (selected), 'Specs', 'API Proxies', 'Shared Flows', 'Offline Trace', and 'API BaaS'. The main area displays the configuration for 'API Proxies > shop > Develop > 1'. At the top, there are buttons for 'Project', 'Save' (highlighted with a red box), 'Revision 1', 'Tools', and 'Deployment'. Below this is a 'Navigator' pane showing a tree view of the configuration: 'shop' (expanded) contains 'Policies' (expanded) with 'Flow Callout-1', 'Flow Callout-2', 'Flow Callout-3', 'PI Request Flow Callout', 'PI Response Flow Callout', and 'Response Flow Callout'; 'Proxy Endpoints' (expanded) with 'default' (expanded) containing 'PreFlow' and 'PostFlow'; 'Target Endpoints' (expanded) with 'default' (expanded) containing 'PreFlow' and 'PostFlow'; and 'Scripts'. The main editor area shows a 'Flow: PreFlow' diagram with a 'Flow Callout-2' rule highlighted by a red box. A 'Step' button is visible below the diagram.

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>
```



LL [Profile Name] [Avatar] [Dropdown Arrow]


**< DEVELOP**

Specs

**API Proxies**

Shared Flows

Offline Trace

API BaaS 

# API Proxies > shop

Project [Dropdown] **Save** Revision 1 [Dropdown]

## Navigator

shop

### ▼ Policies

-  PI Request Flow Callout
-  PI Response Flow Callout

### ▼ Proxy Endpoints

#### ▼ default

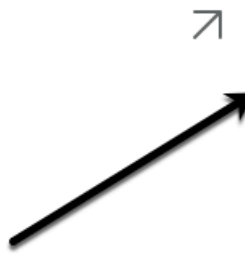
- All** PreFlow
- All** PostFlow

### ▼ Target Endpoints

#### ▼ default

- All** PreFlow
- All** PostFlow

### ▼ Scripts



**9. Save the Revision:**

### Save Revision

Saving the revision will update all deployments, including the deployment in the prod environment. Are you sure you want to save the revision, changing the prod deployment?

Cancel
Save

**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/undeploy.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 50
- [Define an Inline API JSON configuration file](#) on page 83

For more information on ASE sideband deployment, see [Sideband ASE](#) on page 42.


## AWS API gateway integration

---

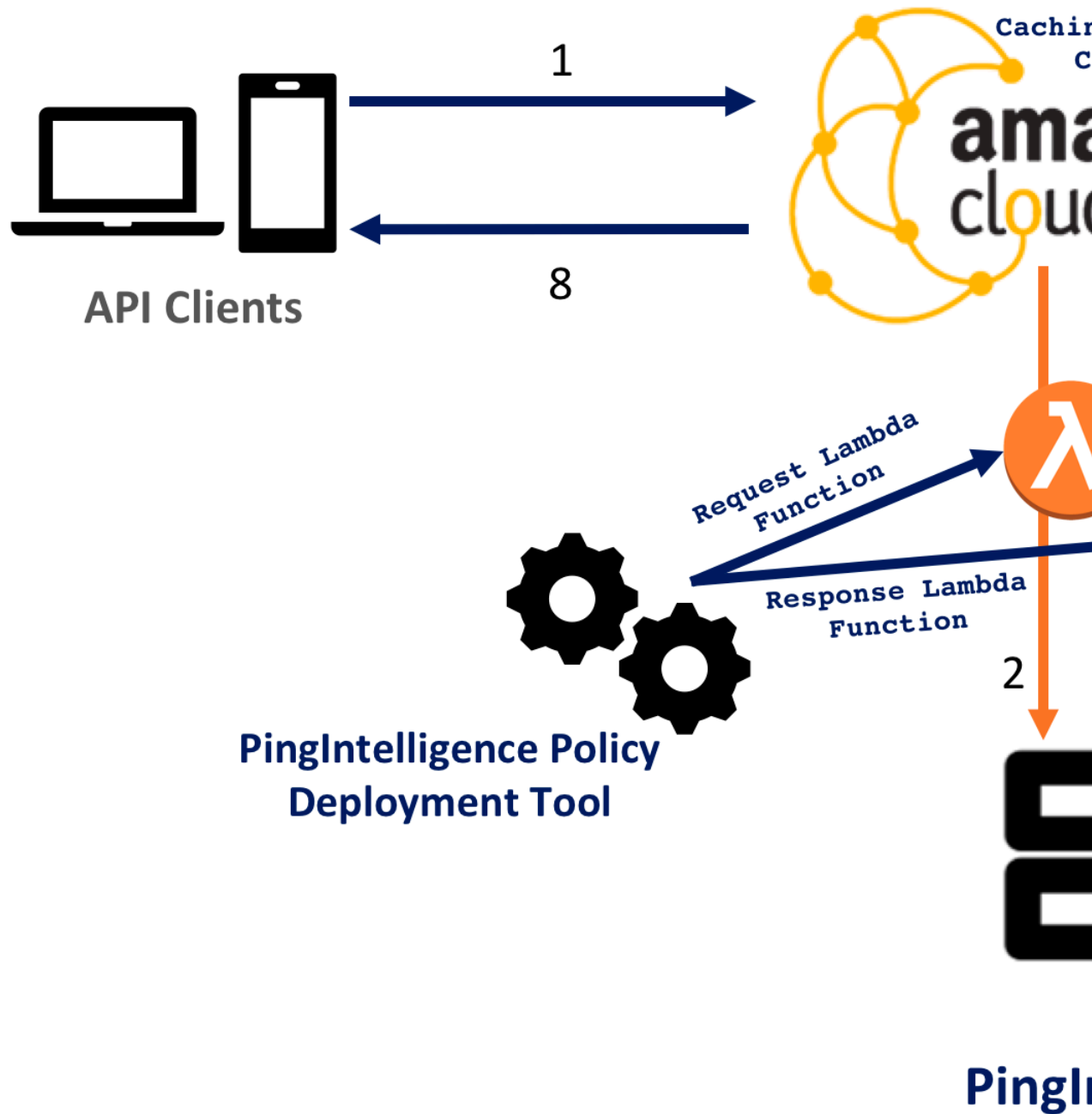
### PingIntelligence AWS API Gateway Integration

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

## Maximum TTL

## Default TTL

## 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: <ul style="list-style-type: none"> <li>▪ <code>deploy.sh</code>: The script to deploy the PingIntelligence policy.</li> <li>▪ <code>undeploy.sh</code>: The script to undeploy the PingIntelligence policy.</li> <li>▪ <code>status.sh</code>: Reports the deployment status of IAM role and Lambda function.</li> </ul>
lib	Jar files and various dependencies. Do not edit the contents of this directory.
policy	Contains the request and response Lambda functions: <ul style="list-style-type: none"> <li>▪ <code>request_lambda.zip</code></li> <li>▪ <code>response_lambda.zip</code></li> </ul>
config	Contains the <code>aws.properties</code> 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 <code>keys</code> and <code>role</code> <p><b>Note:</b> If you running the PingIntelligence policy tool from your local machine, use the <code>keys</code> mode. If you are running the tool from an EC2 instance, use the <code>rolemode</code>.</p>
access_key	AWS access key. This is applicable when the mode is set to <code>keys</code>
secret_key	AWS secret key. This is applicable when the mode is set to <code>keys</code>
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 <a href="#">AWS Lambda Pricing</a>
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	<p>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.</p> <p><b>Note:</b> This field cannot be left blank. In a testing environment, enter the same IP address for primary and secondary ASE host.</p> <p>If both the ASE hosts are unreachable, the request is directly sent to the backend API server.</p>
ase_ssl	<p>Enable or disable SSL communication between Lambda functions and ASE. The default value is true.</p>
ase_sideband_token	<p>Enter the ASE token generated during the <a href="#">prerequisite</a> step.</p>

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:

The screenshot shows the AWS IAM console 'Create role' wizard. The first step, 'Select type of trusted entity', has 'AWS service' selected. The second step, 'Choose the service that will use this role', has 'EC2' selected. The 'Next: Permissions' button is highlighted in blue.

**Create role** 1 2 3 4

Select type of trusted entity

**AWS service**  
EC2, Lambda and others

**Another AWS account**  
Belonging to you or 3rd party

**Web identity**  
Cognito or any OpenID provider

**SAML 2.0 federation**  
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**Lambda**  
Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeBuild	EKS	Lambda	SMS
AWS Backup	CodeDeploy	EMR	Lex	SNS
AWS Support	Config	ElastiCache	License Manager	SWF
Amplify	Connect	Elastic Beanstalk	Machine Learning	SageMaker
AppSync	DMS	Elastic Container Service	Macie	Security Hub
Application Auto Scaling	Data Lifecycle Manager	Elastic Transcoder	MediaConvert	Service Catalog
Application Discovery Service	Data Pipeline	ElasticLoadBalancing	OpsWorks	Step Functions
Auto Scaling	DataSync	Glue	RAM	Storage Gateway
Batch	DeepLens	Greengrass	RDS	Transfer
CloudFormation	Directory Service	GuardDuty	Redshift	Trusted Advisor
CloudHSM	DynamoDB	Inspector	Rekognition	VPC
CloudTrail	EC2	IoT	S3	WorkLink
CloudWatch Events	EC2 - Fleet	Kinesis		

\* Required

Cancel **Next: Permissions**

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**.

aws
Services ▾
Resource Groups ▾
🔔

## Create role

1
2
3
4

### Review

Provide the required information below and review this role before you create it.

**Role name\***

Use alphanumeric and '+,=,@-\_' characters. Maximum 64 characters.

**Role description**

Maximum 1000 characters. Use alphanumeric and '+,=,@-\_' characters.

**Trusted entities** AWS service: ec2.amazonaws.com

**Policies**

- 📦 [IAMFullAccess](#) ↗
- 📦 [CloudFrontFullAccess](#) ↗
- 📦 [AWSLambdaFullAccess](#) ↗
- 📦 [AmazonEC2FullAccess](#) ↗

**Permissions boundary** Permissions boundary is not set

The new role will receive the following tag

Key	Value
PI	PIAWS

**\* Required**

Cancel
Previous
Create role

- 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:

The screenshot shows the AWS IAM console interface. At the top, there is a navigation bar with the AWS logo, 'Services', and 'Resource Groups'. On the left, a sidebar contains a search bar labeled 'Search IAM' and a list of navigation items: Dashboard, Groups, Users, Roles (highlighted with an orange bar), Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area shows the breadcrumb 'Roles > PIDeploymentToolRole' and the title 'Summary'. Below the title, there are three tabs: 'Permissions', 'Trust relationships' (selected with an orange underline), and 'Tags'. The 'Trust relationships' tab content includes the text 'You can view the trusted entities that can assume t', a blue button labeled 'Edit trust relationship', and a section titled 'Trusted entities' with the text 'The following trusted entities can assume this role.'. Below this, a yellow highlighted box contains the heading 'Trusted entities' and two entries: 'The identity provider(s) ec2.amazonaws.com' and 'The identity provider(s) lambda.amazonaws.com'.

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"
    }
  ]
}
```

```
}
```





Services ▾

Resource Groups ▾



## Edit Trust Relationship

## Edit Trust R

You can customize trust

### 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,

The screenshot shows the AWS console interface for configuring an instance. The top navigation bar includes the AWS logo, 'Services', and 'Resource Groups'. Below the navigation bar are three steps: '1. Choose AMI', '2. Choose Instance Type', and '3. Configure Instance Details', with the third step being the active one. The main heading is 'Step 3: Configure Instance Details', followed by the instruction 'Configure the instance to suit your requirements. You can launch'. The configuration form consists of several rows, each with a label, an information icon, and a value field. The 'IAM role' field is highlighted with a blue border and contains the text 'PIDeployme'. Other visible fields include 'Number of instances' (1), 'Purchasing option' (Request Spot), 'Network' (vpc-5e1115), 'Subnet' (No preference), 'Auto-assign Public IP' (Use subnet), 'Placement group' (Add instance), 'Capacity Reservation' (Open), 'Shutdown behavior' (Stop), 'Enable termination protection' (Protect against accidental termination), 'Monitoring' (Enable CloudWatch), and 'Tenancy' (Shared - Run on shared hardware).

Number of instances	<a href="#">i</a>	1
Purchasing option	<a href="#">i</a>	<input type="checkbox"/> Request Spot
Network	<a href="#">i</a>	vpc-5e1115
Subnet	<a href="#">i</a>	No preference
Auto-assign Public IP	<a href="#">i</a>	Use subnet
Placement group	<a href="#">i</a>	<input type="checkbox"/> Add instance
Capacity Reservation	<a href="#">i</a>	Open
IAM role	<a href="#">i</a>	<b>PIDeployme</b>
Shutdown behavior	<a href="#">i</a>	Stop
Enable termination protection	<a href="#">i</a>	<input type="checkbox"/> Protect against accidental termination
Monitoring	<a href="#">i</a>	<input type="checkbox"/> Enable CloudWatch <a href="#">Additional configuration</a>
Tenancy	<a href="#">i</a>	Shared - Run on shared hardware <a href="#">Additional configuration</a>
Elastic Inference	<a href="#">i</a>	<input type="checkbox"/> Add an Elastic Inference Accelerator <a href="#">Additional configuration</a>

## 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 Lambda 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.
```

## 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.](#)

## API discovery

APIs can be automatically discovered using the PingIntelligence ABS AI Engine. For more information on enabling discovery. For more information on enabling discovery, see [Enable and disable discovery](#). APIs are discovered when a global API JSON file is added to PingIntelligence ASE. For more information, see [API discovery](#). After the APIs are discovered by ABS, AAD adds the API JSON to ASE. Install AAD to add discovered APIs to ASE.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD logs
- `util` - Contains the `check_ports.sh`. Run on the machine with the AAD tool to check if ASE and ABS default ports are open.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	NA
<code>aad.env.dev.fullsync</code>	NA
<code>aad.mode</code>	Set the value to <code>discovery</code> for AAD to work in discovery mode.
<code>abs.host</code>	ABS host IP address
<code>abs.port</code>	ABS host port number
<code>abs.access_key</code>	ABS access key
<code>abs.secret_key</code>	ABS secret key
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	Set to true for ABS-AAD communication to use SSL.
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>

abs.query.interval	NA.
aad.log.level	The log level of AAD log files. The default value is INFO. Other possible values are: ALL<DEBUG<INFO<WARN<ERROR<FATAL<OFF
gateway.management.url	NA
gateway.management.username	NA
gateway.management.password	NA
pingaccess.management.url	NA
pingaccess.management.username	NA
pingaccess.management.password	NA

Following is a sample aad.properties file:

```
# Automated API Discovery (AAD)
# Automated API Discovery (AAD)

# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod

# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
# or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E

### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
```

```
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsjTC+lxddGqv3XUUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSmv5mZllw3WpAsj1oPF3d5Et170Y=
# ABS secret key
abs.secret_key=OBF:AES:RsjTC/tx/sp
+7XXtr8+1rnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74IahxfEgp:dLfcNugFUCcfsq1nBXQzflTvwAWiPit8ulseHxi+Z0tk=

### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+1pEqcKQnFG/UN3Ezfz0DMa/
kmI=:Az82rlUFftMGpMxF7unelJZUucX191102QgKvHD36vU=
```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `gateway.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

### Generate `aad_master.key`

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```
opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
  config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
  abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start AAD after passwords are obfuscated.

Start AAD

#### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/stop.sh
Ping Identity Inc.
AAD is stopped.
```

## 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.

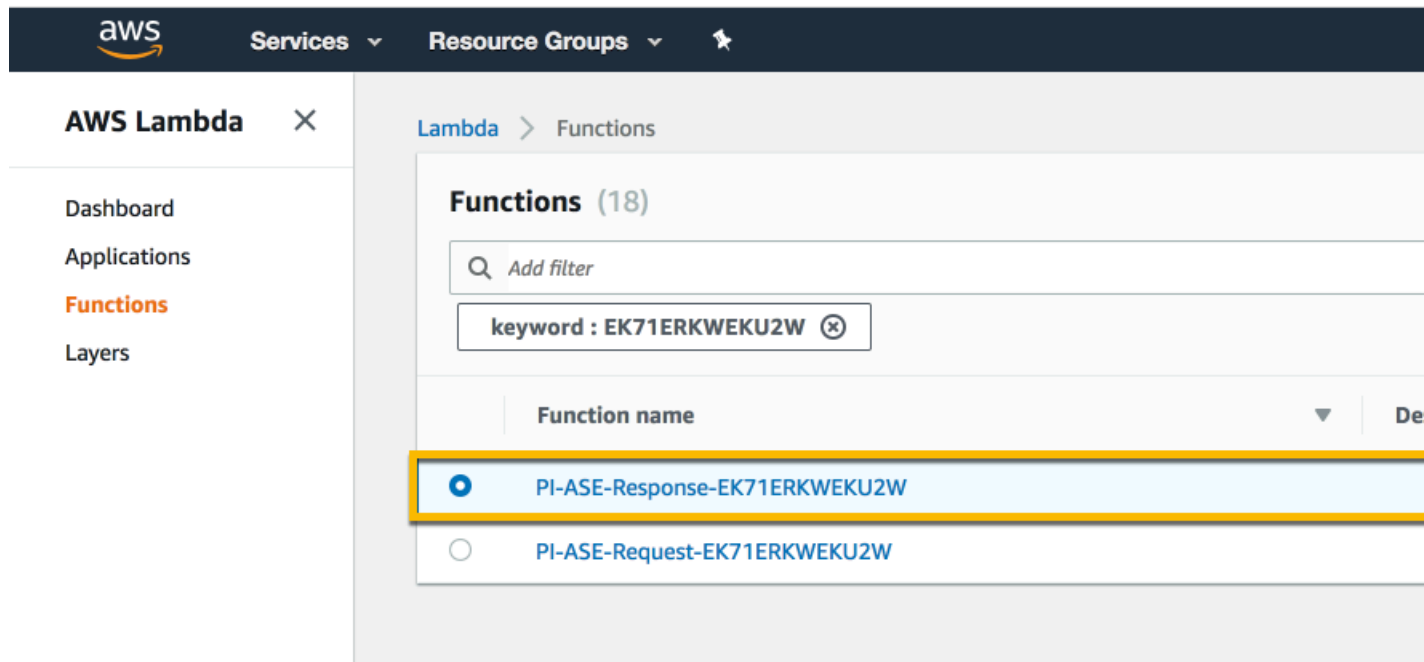


Check the progress of detaching the policy from the CloudFront in the AWS GUI as shown in the following screenshot:

The screenshot shows the AWS Management Console interface for CloudFront Distributions. The left sidebar contains navigation options under 'Distributions', 'Reports & Analytics', and 'Security'. The main content area displays a 'Create Distribution' button and a 'Viewing:' dropdown menu set to 'Any Delivery Method'. Below this is a table of distributions with a 'Delivery Method' column. The first row is selected, indicated by a blue checkmark in the checkbox and a yellow highlight around the row.

	Delivery Method	
<input type="checkbox"/>	Web	
<input type="checkbox"/>	Web	
<input type="checkbox"/>	Web	
<input type="checkbox"/>	Web	
<input type="checkbox"/>	Web	
<input checked="" type="checkbox"/>	Web	
<input type="checkbox"/>	Web	

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.



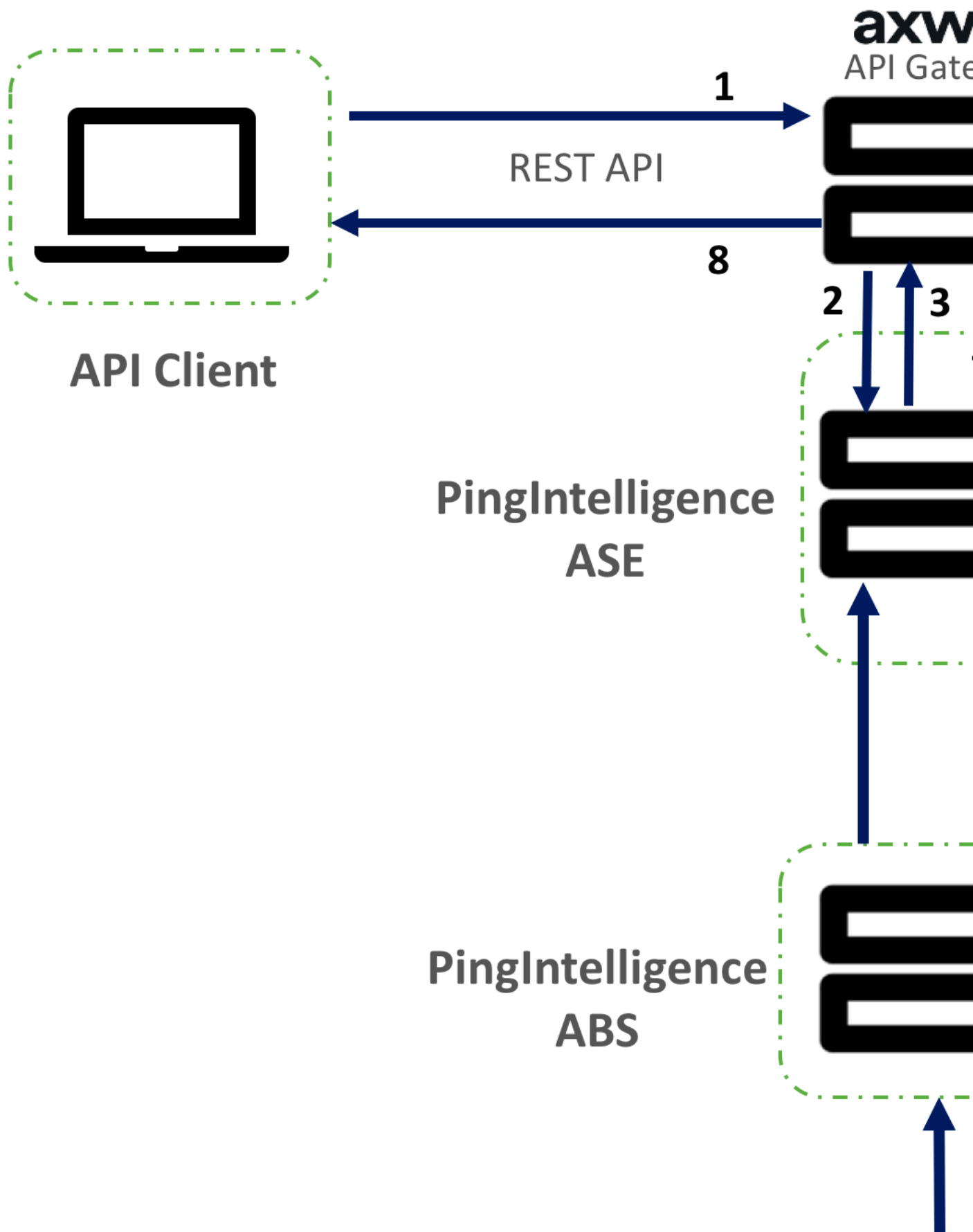
**Note:** If the Lambda functions are not deleted, then the following message is displayed on the console: Deletion of the Lambda function may take up to one hour. Please re-run `undeploy.sh` after one hour.

## 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 and

#### Editing API, My API

Editing virtualized API. Make your changes and click "Save" to commit them, or "Cancel" to quit.

← Save   Apply   Cancel

Inbound   Outbound   API   API Methods   Security Profiles   Authentication Profiles   CORS Profiles   Trusted Certificates



### OAuth Security Device ×

General
Authorization
Grant Type: Implicit
Grant Type: Authorization Code

**\*Access token store:** OAuth Access Token Store ▼

**\*Scopes must match:** Any ▼

**\*Scopes:** resource.WRITE, resource.READ ?

Remove credentials on success:

OK
Cancel

7.6.2.

- **OAuth token store:** If you wish to detect username based attacks, make sure that OAuth token store is configured in Axway.

i **Note:** PingIntelligence only supports Axway's bundled authorization server for username based attack detection.

- **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.

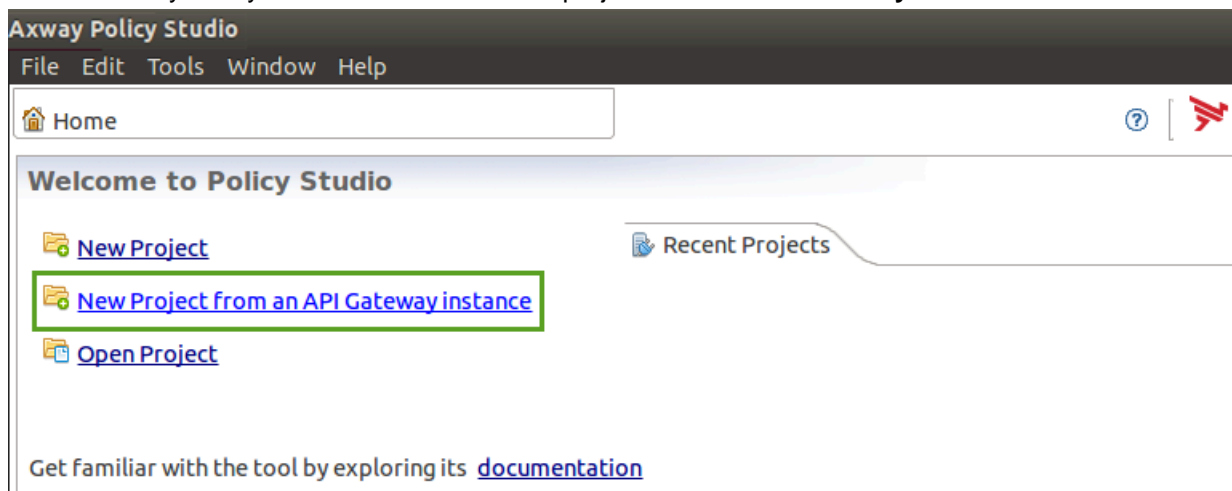
### Deploy PingIntelligence policy

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**:

The screenshot shows the 'New Project' dialog box in Axway Policy Studio. The main window title is 'Axway Policy Studio' with a menu bar containing 'File', 'Edit', 'Tools', 'Window', and 'Help'. Below the menu bar is a 'Home' button. The main content area is titled 'Welcome to Policy Studio' and contains three links: 'New Project', 'New Project from an API Gateway instance', and 'Open Project'. The 'New Project' dialog box is open, showing the 'Open connection' section. It prompts the user to 'Specify the destination you want to connect to'. Under 'Saved Sessions', there is a dropdown menu with 'Admin Node Manager - localhost' selected. The 'Connection Details' section has three input fields: 'Host' with 'vortex-135', 'Username' with 'admin', and 'Password' with '\*\*\*\*\*'. Below this is an 'Advanced' section with a right-pointing arrow. At the bottom of the dialog box are three buttons: 'Help', '< Back', and 'Next >'.

Axway Policy Studio

File Edit Tools Window Help

Home

Welcome to Policy Studio

- [New Project](#)
- [New Project from an API Gateway instance](#)
- [Open Project](#)

New Project

Open connection

Specify the destination you want to connect to

Saved Sessions

Session: Admin Node Manager - localhost

Connection Details

Host: vortex-135

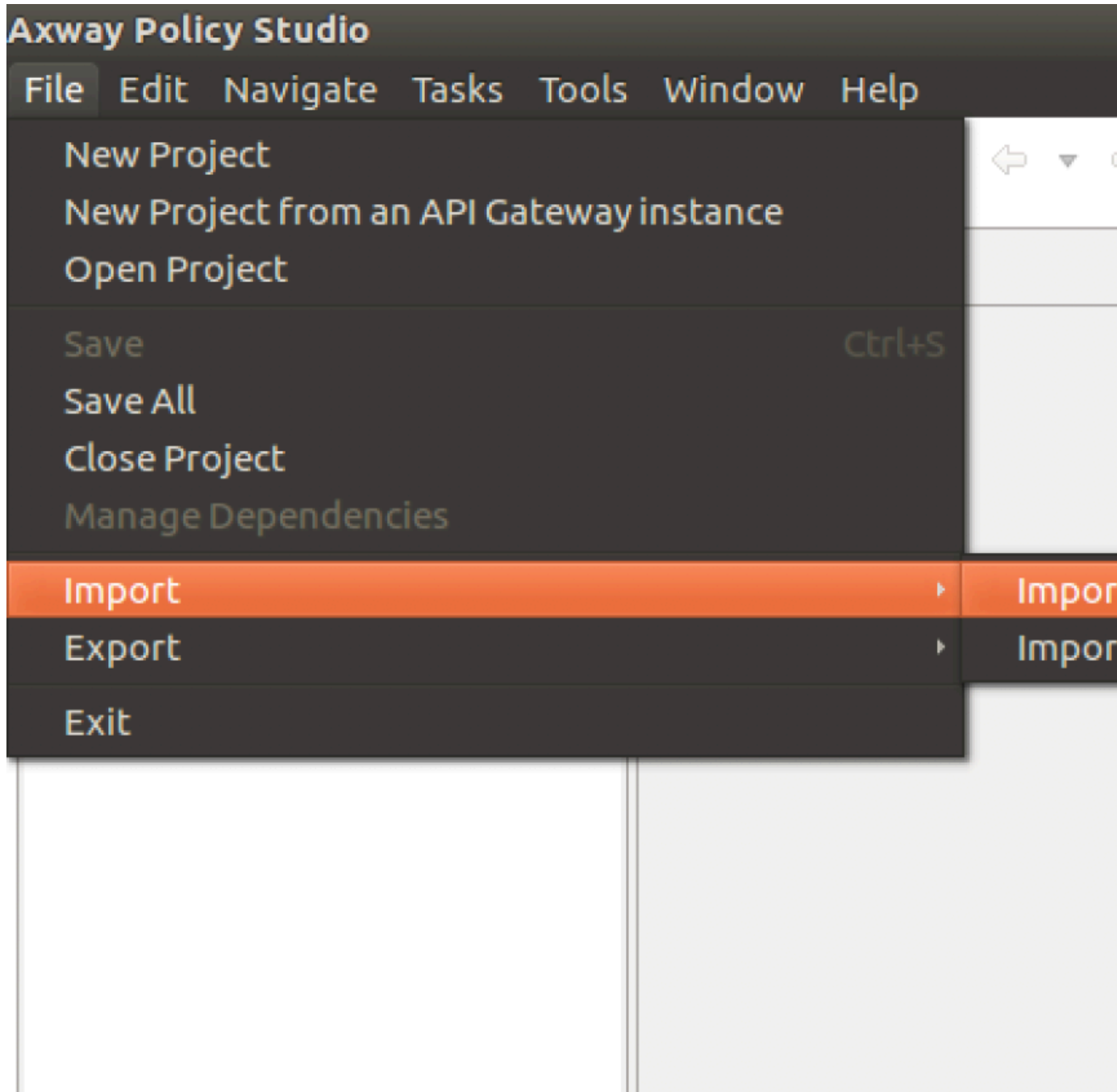
Username: admin

Password: \*\*\*\*\*

Advanced

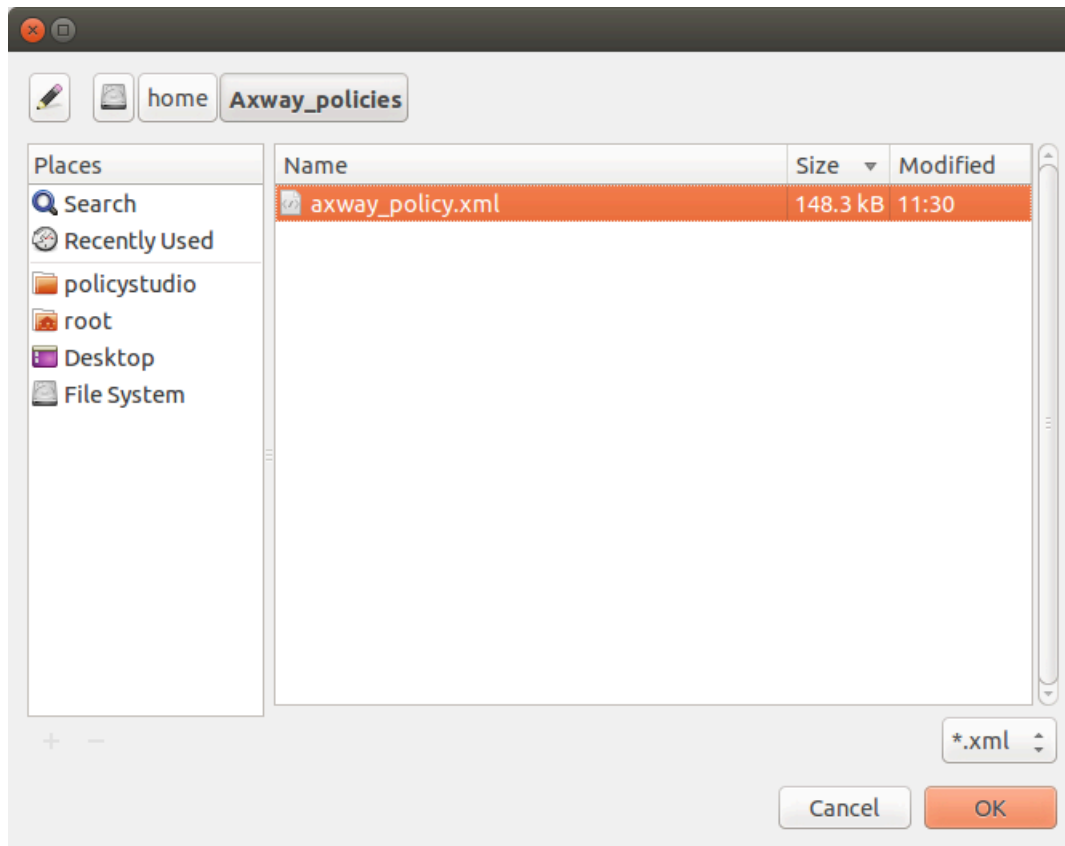
Help < Back Next >

4. Click **Import configuration fragment** from the **File** sub menu in the menu bar



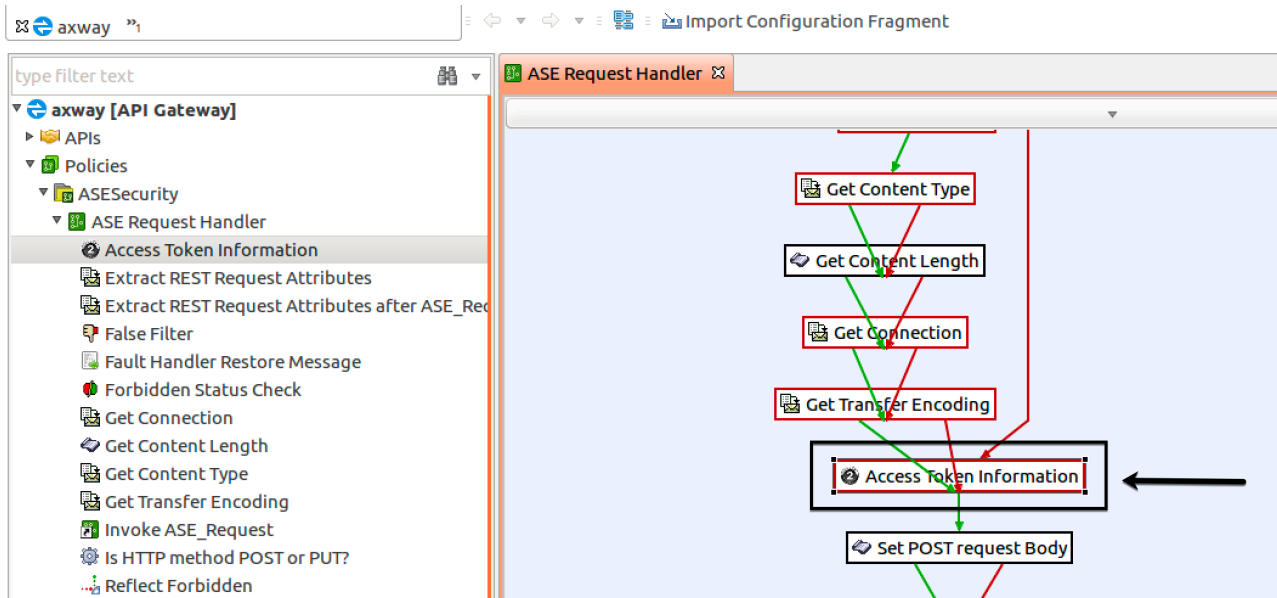
From the pop-up window, import the Axway Policy from the directory where it was saved. Select the policy and click **OK**:



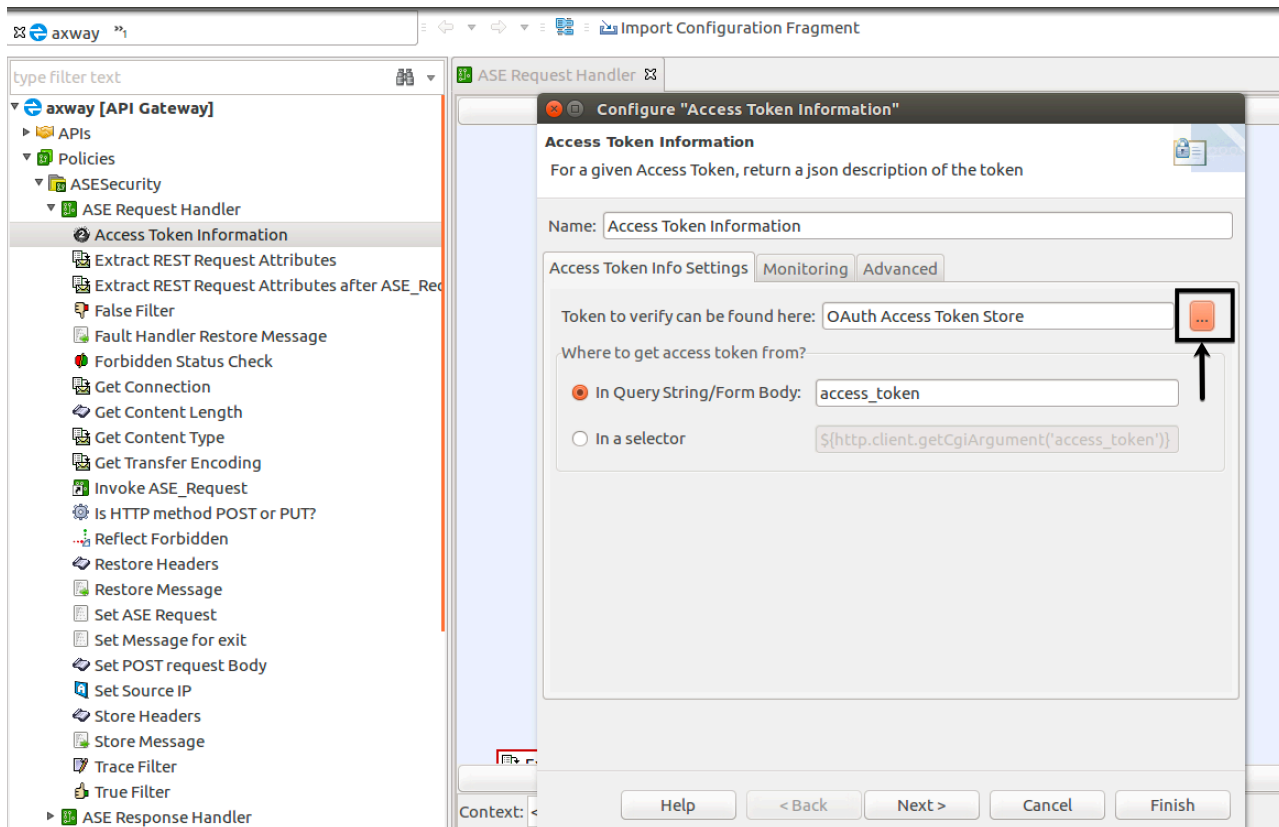


5. After the Axway Policy is imported, click on **Policies > ASESecurity > ASE Request Handler > Access Token Information**. Double click on

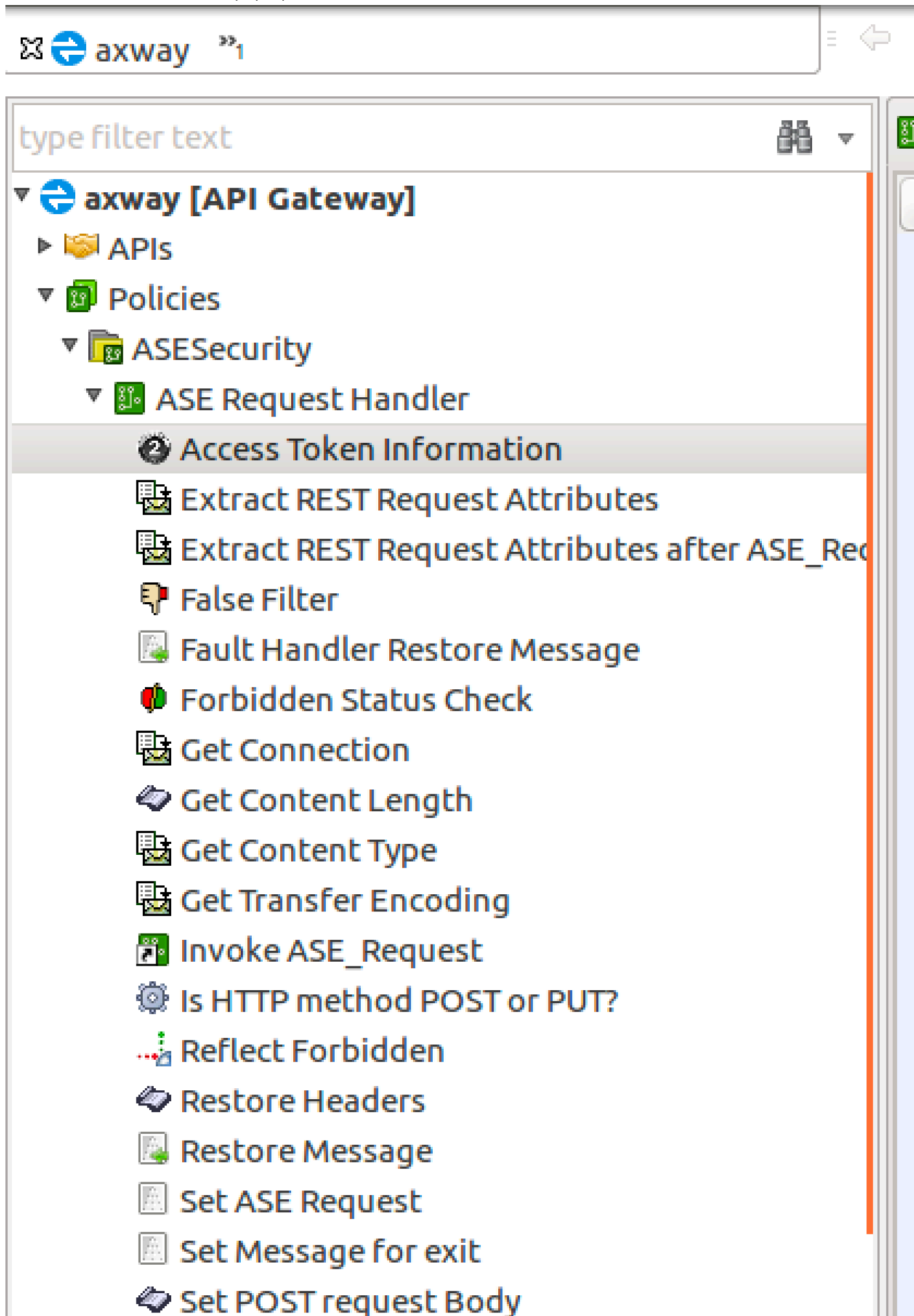
Access Token Information box in the ASE Request Handler window.



- a. In the **Configure "Access Token Information"** pop-up window, enter your OAuth token store information and click the ... button.



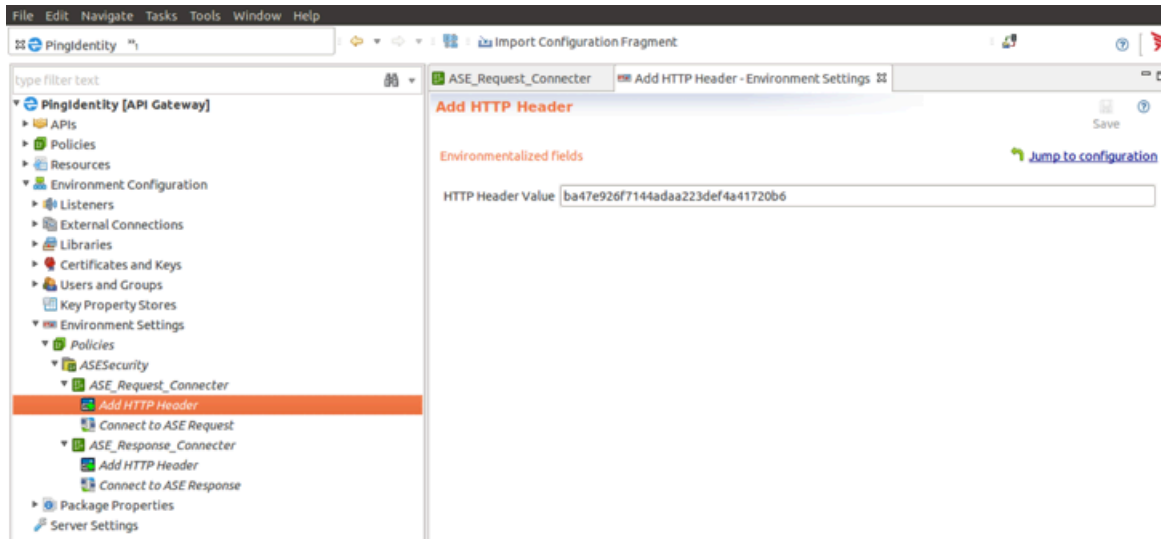
- b. In the **Select OAuth Cache** pop-up window, select the OAuth token store.



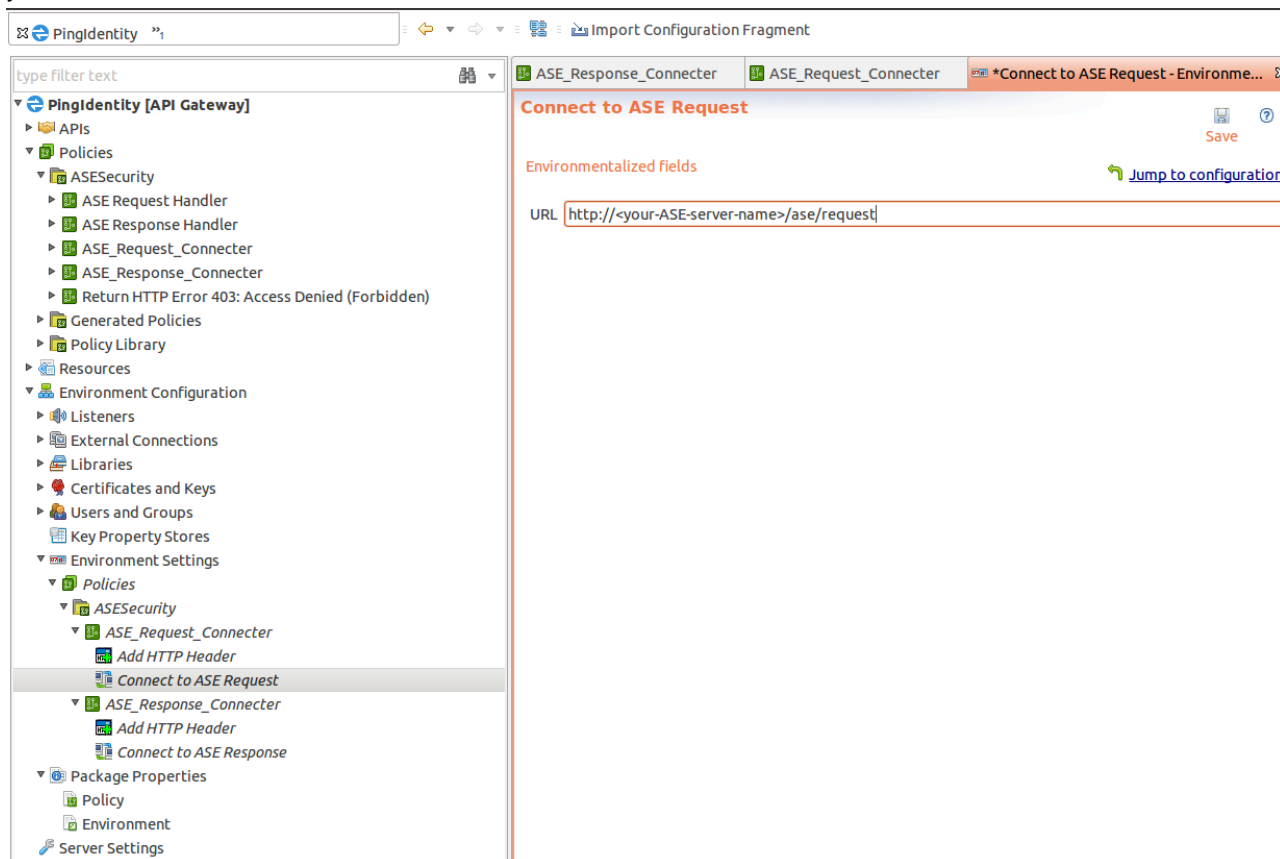
The screenshot shows a web browser window with the URL 'axway'. Below the browser, a search bar contains the text 'type filter text'. A tree view on the left side of the console shows the following structure:

- axway [API Gateway]
  - APIs
  - Policies
    - ASESecurity
      - ASE Request Handler
        - Access Token Information** (selected)
        - Extract REST Request Attributes
        - Extract REST Request Attributes after ASE\_Rec
        - False Filter
        - Fault Handler Restore Message
        - Forbidden Status Check
        - Get Connection
        - Get Content Length
        - Get Content Type
        - Get Transfer Encoding
        - Invoke ASE\_Request
        - Is HTTP method POST or PUT?
        - Reflect Forbidden
        - Restore Headers
        - Restore Message
        - Set ASE Request
        - Set Message for exit
        - Set POST request Body

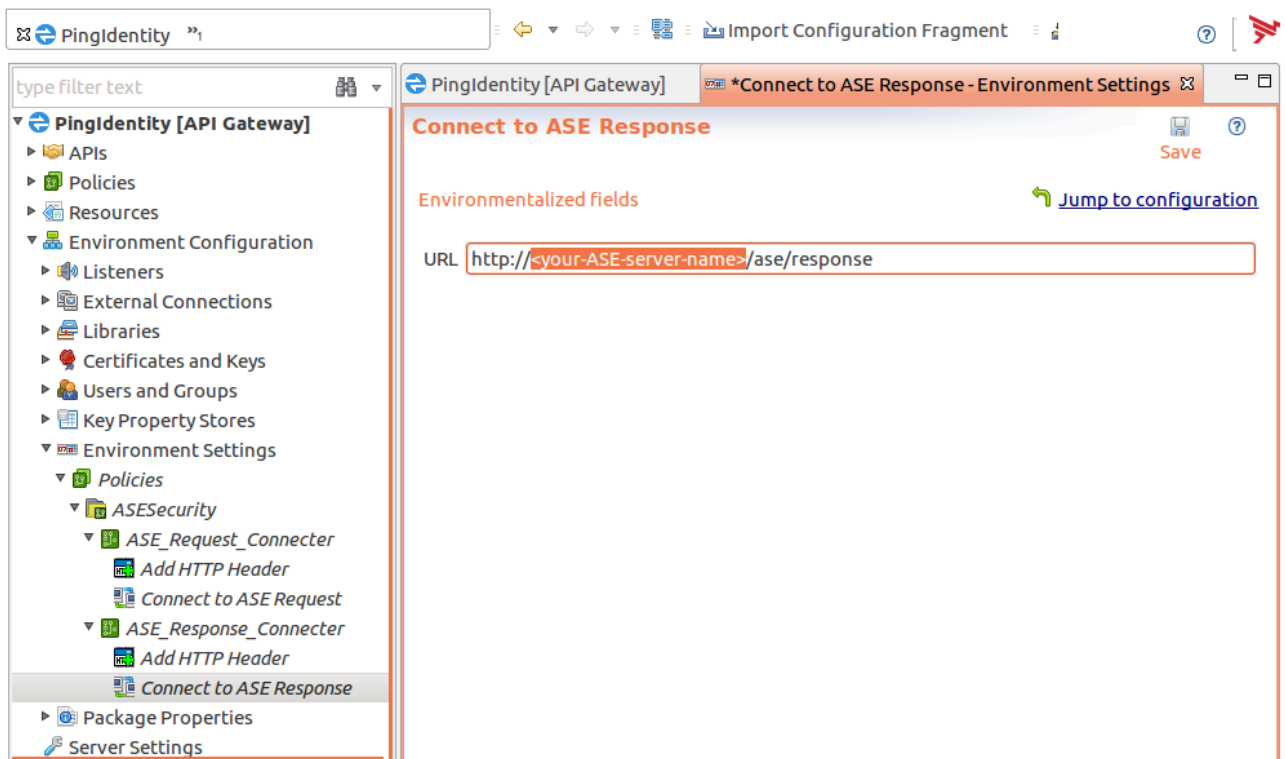
6. After the Axway Policy is imported, click **Environment Settings** in the left-hand column and Click **Add HTTP Header**. In the HTTP Header Value field, enter the ASE authentication token that was created.



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:

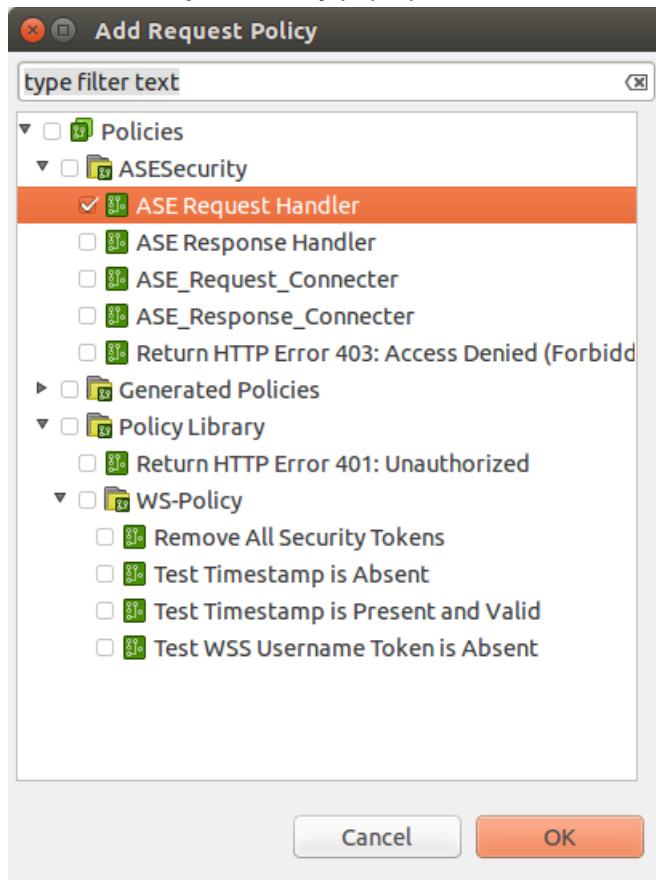


8. In the **Environment Settings** in the left-hand column, click **Connect to ASE Response** under **ASE\_Response\_Connector**. Enter the IP address or the hostname of your ASE in the **URL** field as shown in the screen shot:

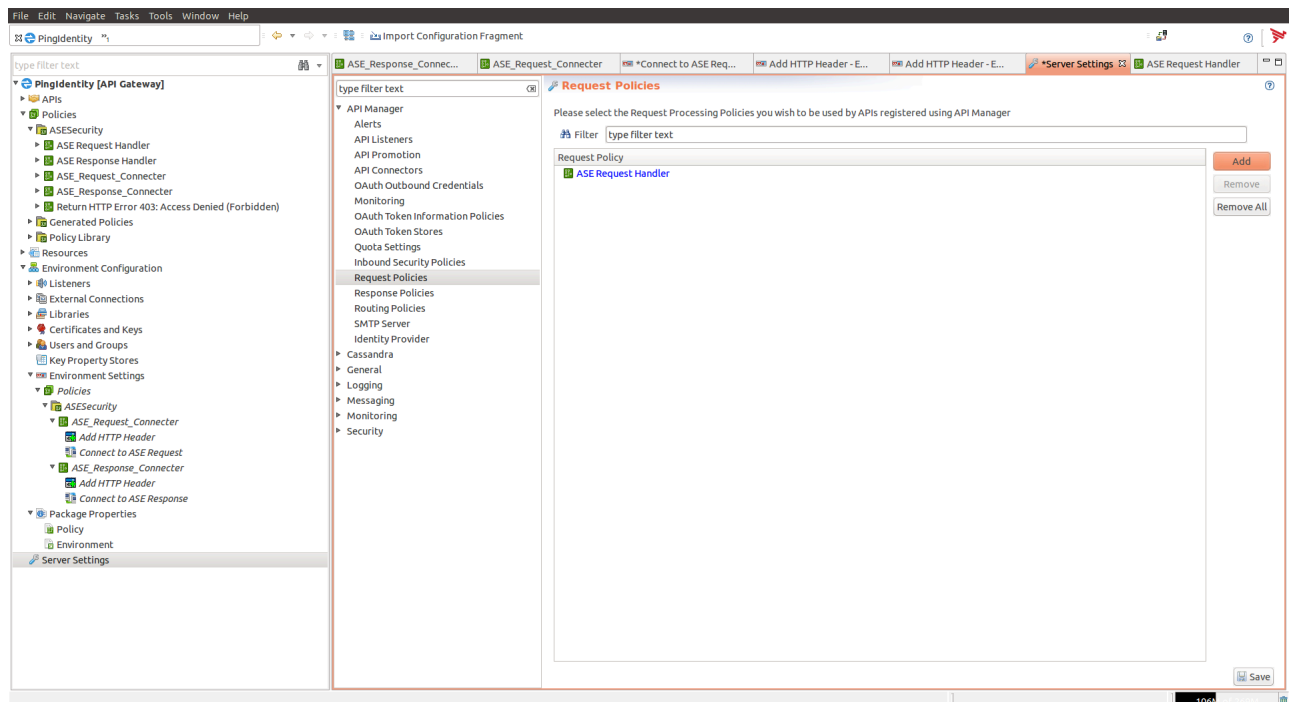


9. In the left pane of the window, click **Server Settings**.
10. In the Server Settings window, double-click **Request Policies** under **API Manager**

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

The screenshot displays the Axway API Manager configuration interface. The left pane shows a tree view of the configuration for 'pi [API Gateway]'. The right pane shows the 'Remote host settings' for the selected listener '34.212.173.5 : 8000'. The 'Remote host settings' dialog box is open, showing fields for 'Host alias', 'Host name', and 'Port', along with checkboxes for 'Allow HTTP', 'Include CORS', 'Send Server', and 'Verify server'.

### 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**

The screenshot shows the 'Manage frontend API' interface in Axway API Manager. On the left, there is a sidebar with 'API' selected, and sub-items for 'Frontend API', 'Backend API', and 'API Catalog'. The main area displays a table of APIs:

Name	Version	Service Type	Organization	Frontend URL	State	Tags	Created on
decoy	1.0	REST	API Development	https://172.17.0.1:8065/decoy	Published		8 March 2018 00:19
demoshop	1.0	REST	API Development	https://172.17.0.1:8065/app	Unpublished		9 March 2018 01:11
shop	1.0	REST	API Development	https://shopapi/shopapi	Published		7 March 2018 23:04
shop-books	1.0	REST	API Development	https://books/shopapi-books	Published		8 March 2018 00:18
shop-electronics	1.0	REST	API Development	https://172.17.0.1:8065/shopapi-electronics	Unpublished		8 March 2018 00:19

Below the table is a 'Create new API' dialog box with a dropdown menu labeled '\*Select existing backend API:' and 'OK' and 'Cancel' buttons.

3. Click **Outbound** tab and enter Backend Service URL (your backend application server) and Request

The screenshot shows the 'Editing API, test' interface in Axway API Manager. The 'Outbound' tab is selected, and the configuration for the API Gateway is displayed. The configuration includes:

- Outbound authentication profile:** No authentication
- Backend service URL:** http://vortex-136:2200
- Request policy:** ASE Request Handler
- Default method routing:** API Proxy
- Response policy:** ASE Response Handler

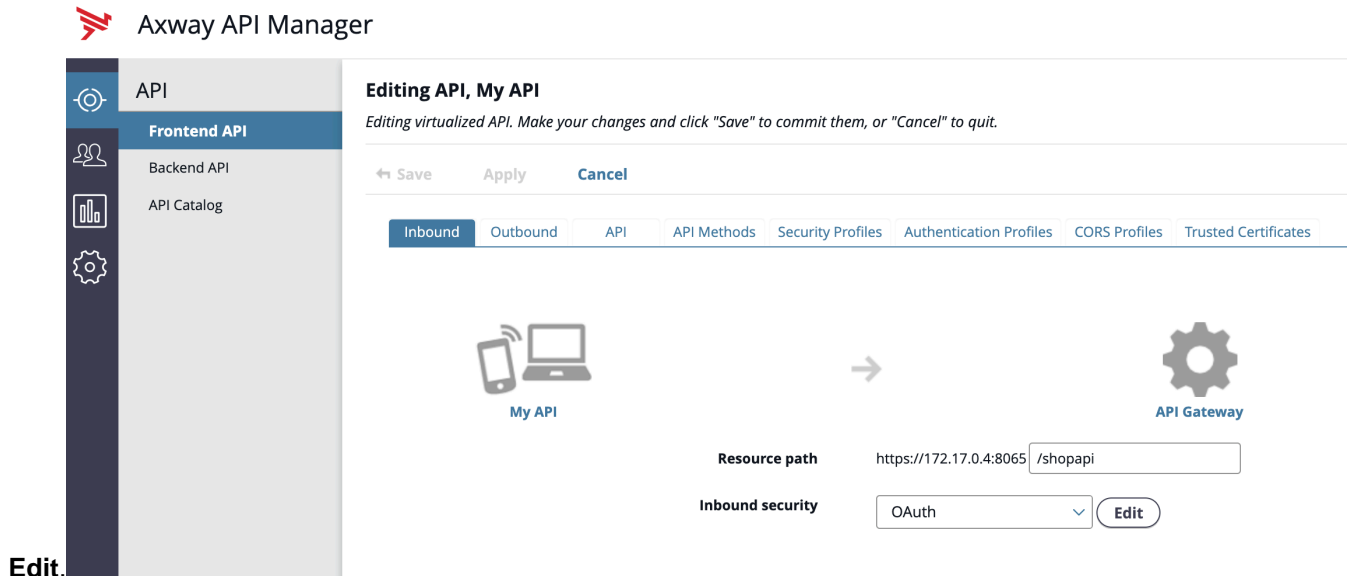
The configuration is highlighted with a green box. Below the configuration, there is a section for 'PER-METHOD OVERRIDE'.

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



**Axway API Manager**

**Editing API, My API**  
Editing virtualized API. Make your changes and click "Save" to commit them, or "Cancel" to quit.

← Save Apply Cancel

Inbound Outbound API API Methods Security Profiles Authentication Profiles CORS Profiles Trusted Certificates

My API → API Gateway


Resource path:

Inbound security:

Edit.

3. In the **OAuth Security Device** window, disable **Remove credentials on success** radio

### OAuth Security Device



General Authorization Grant Type: Implicit Grant Type: Authorization Code

\*Access token store:

\*Scopes must match:

\*Scopes:  ?

Remove credentials on success:

button.

### API discovery

Install and configure AAD to automatically capture API definitions from your Axway API gateway. AAD discovers these API definitions, converts them into PingIntelligence API JSON files, and adds the definitions to ASE.

AAD supports two operating modes:

- Dev – the PingAccess API definitions are mirrored in the PingIntelligence environment. If an API is removed from PingAccess, it is deleted from PingIntelligence.
- Prod - the PingAccess API definitions are mirrored in the PingIntelligence environment but APIs are not deleted from PingIntelligence.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxvf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD's logs
- `util` - Contains the `check_ports.sh`. Run on the machine with the AAD tool to check ASE and ABS default ports.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	<p>Set the key to <code>dev</code> if you are fetching API definitions in an API development environment.</p> <p>Set the key to <code>prod</code> if you are fetching API definitions in a production environment</p>
<code>aad.env.dev.fullsync</code>	<p>Set to <code>true</code> to mirror the PingAccess API definitions in PingIntelligence. AAD deletes any APIs from PingIntelligence that are removed from PingAccess.</p> <p>Set to <code>false</code> to synchronize PingAccess and PingIntelligence definitions without deleting other APIs present in PingIntelligence.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> <code>aad.env.dev.fullsync</code> key is only valid in a development environment (<code>aad.env=dev</code>)</p> </div>
<code>aad.mode</code>	<p>Set the value to <code>gateway</code> when ASE is deployed in sideband mode.</p> <p>For more information on ASE modes, see the <i>ASE Admin Guide</i>.</p>
<code>abs.host</code>	NA
<code>abs.port</code>	NA
<code>abs.access_key</code>	NA
<code>abs.secret_key</code>	NA
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	NA
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>
<code>abs.query.interval</code>	NA.
<code>aad.log.level</code>	The log level of AAD log files. The default value is <code>INFO</code> . Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code>

gateway.management.url	URL of the Axway API Gateway. Only valid when aad.mode is gateway.
gateway.management.username	Username to connect to the API Gateway. Only valid when aad.mode is gateway.
gateway.management.password	Password to connect to the API Gateway. Only valid when aad.mode is gateway.
pingaccess.management.url	NA
pingaccess.management.username	NA
pingaccess.management.password	NA

Following is a sample aad.properties file:

```
# Automated API Discovery (AAD)
# AAD mode. Valid values discovery, span_port, and axway
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=gateway
# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwEL2rW7+:v7j6VGWaoMjUNcc4IMaTOMtLL8hUPOLWrq7
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwEL2rW7+:v7j6VGWaoMjUNcc4IMaTOMtLL8hUPOLWrq7
### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
#abs.host=127.0.0.1
# ABS management port
#abs.port=8080
# ABS SSL enabled ( true or false )
#abs.ssl=true
# ABS access key
#abs.access_key=OBF:AES:Rs7jTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSvm5mZllw3WpAsj1oPF3d5Et170Y=
# ABS secret key
#abs.secret_key=OBF:AES:Rs7jTC/tx/sp
+7XXtr8+lrnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74LahxfEgp:dLfcNugFUCcfsglnBXQzflTvAWiPit8ulseHxi+Z0tk=
```

```
### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+1pEqcKQnFG/UN3Ezfz0DMA/
kmI=:Az82rlUFftMGpMxF7unelJZUucX191102QgKvHD36vU=
```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `gateway.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

### Generate `aad_master.key`

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```
opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
```

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

Start [AAD](#) after passwords are obfuscated.

Start AAD

#### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 3.2.1 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/stop.sh
Ping Identity Inc.
AAD is stopped.
```

### Axway API Manager configuration for AAD

AAD pulls the API definition from Axway API Manager and converts them to a JSON format compatible with ASE. AAD needs certain tags to be configured in Axway API Manager for AAD 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 AAD 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 AAD 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 AAD tool 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:



# Axway API Manager



API



**Frontend API**



Backend API



API Catalog

## Viewing API, shop

*The following API is read-only and cannot be modified.*

← Save

Apply

Cancel

Inbound

Outbound

**API**

### General

**Documentation**



### 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`

The screenshot shows the Axway API Manager interface. The left sidebar contains navigation options: API, Frontend API, Backend API, API Catalog, and a settings icon. The main content area is titled 'API' and has tabs for Inbound, Outbound, API, API Methods, Security Profiles, Authentication Profiles, and CORS Profiles. The 'API' tab is active, showing the 'General' section with fields for Image, \*API name (decoy), API version (1.0), Service type (REST), API summary, State (Published), Created By (API Manager Administrator), and Created on (14 May 2018, 17:47). Below this is the 'Documentation' section with a Description dropdown set to 'Use original API description'. The 'Tags' section is highlighted with a green box and contains a table:

Tags:	TAG:	VALUES
	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.

The screenshot shows the navigation menu for an API Gateway environment named 'Apr\_29'. The menu is organized as follows:

- APIs
- Policies
  - Enable-xff
  - Generated Policies
  - OAuth 2.0
  - Policy Library
  - QuickStart
  - Sample Policies
- Resources
- Environment Configuration
- Server Settings

### OAuth2 Token and API Keys

If you have configured the API Key in Request Header or in Query String, AAD 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, AAD 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.

**Axway API Manager**

**Editing API, shop**  
Editing virtualized API. Make your changes and click "Save" to commit them, or "

← Save   Apply   Cancel

Inbound   Outbound   **API**   API Methods   Security Profiles   Authentication

shop

Resource path   https://192.168.11.1

Inbound security   Pass Through

```
{
  "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": "Keyld",
    "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

---

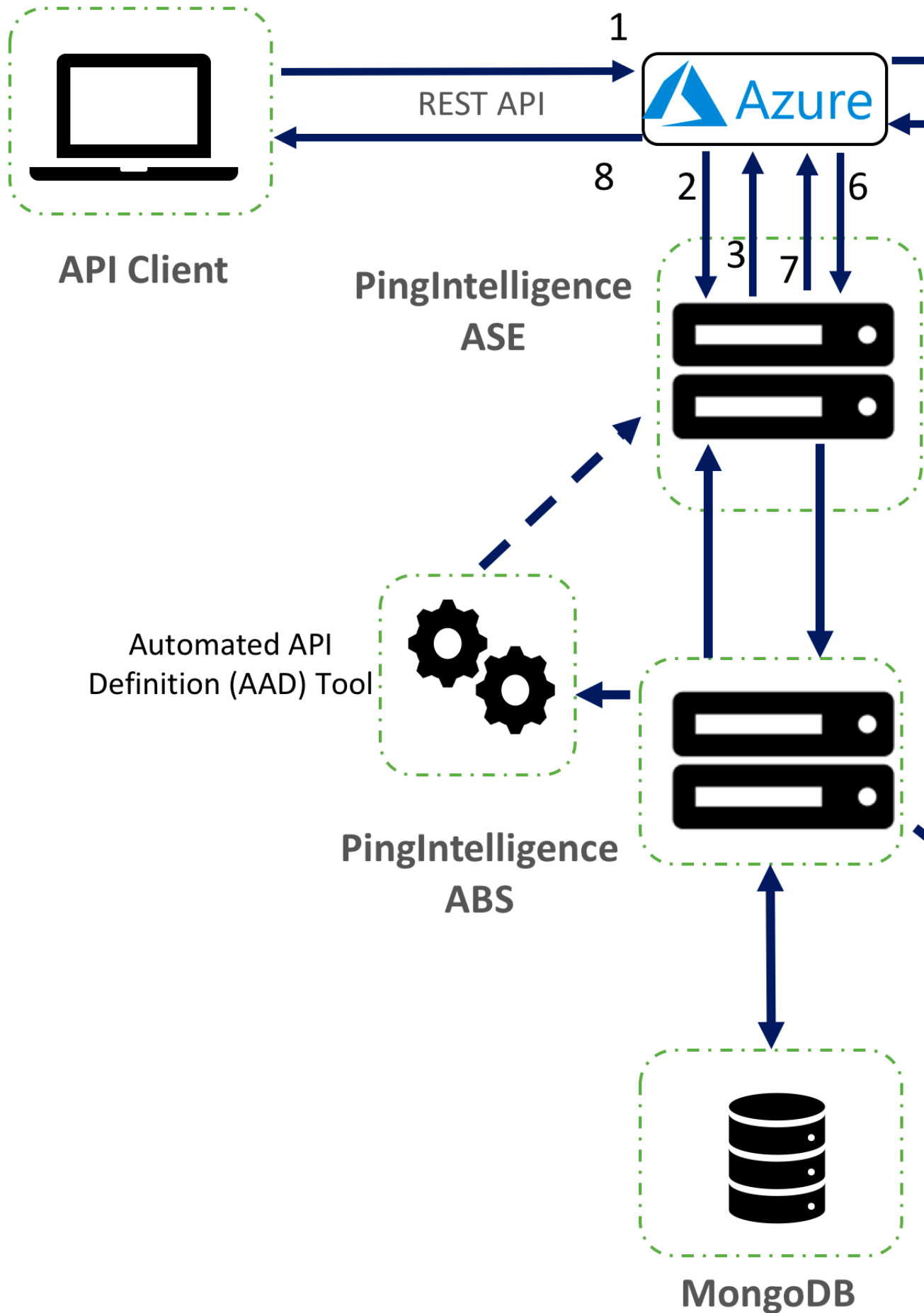
### Azure APIM sideband integration

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:

- **Non-blocking mode:** When the PingIntelligence policy is configured in the non-blocking 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.
- **Blocking mode:** When the PingIntelligence policy is configured in the blocking 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, 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
- 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**

The screenshot displays the Azure portal interface. On the left is a dark navigation pane with various service icons and labels. The main content area on the right features a search bar at the top, followed by a section titled 'API Management' which lists several services. Below that is a section titled 'Security' which lists more services, with 'CA certificates' highlighted in a light blue background.

**Navigation Pane (Left):**

- Home
- Dashboard
- All services
- ★ FAVORITES
- All resources
- Resource groups
- App Services
- Function Apps
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- Storage accounts
- Virtual networks
- Azure Active Directory
- Monitor
- Advisor
- Security Center
- Cost Management + Billing
- Help + support

**Main Content Area (Right):**

Search (Ctrl+ /)

**API Management**

- Quickstart
- APIs
- Products
- Named values
- Tags
- Analytics (preview)
- Users
- Subscriptions
- Groups
- Notifications
- Notification templates
- Issues
- Repository
- Management API

**Security**

- Identities
- OAuth 2.0
- OpenID Connect
- CA certificates
- Client 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                   : 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 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.
- **ase-token**: The authentication token for secure communication between Azure API Management service and ASE
- **blocking**: Set the value to 0 (disable blocking) if you do not want the APIM to wait for a response from PingIntelligence ASE before sending the request to backend server. Set the value to 1 (enable blocking), if you want the Azure gateway to wait for the ASE response before sending the request to the backend server or blocking it.
- **connection-timeout**: The number of seconds for which the API Management Service waits for ASE to respond.
- **oauth2-jwt-username-claim**: JWT claim name for username.
- **retry-count**: The number of times APIM tries to connect to ASE

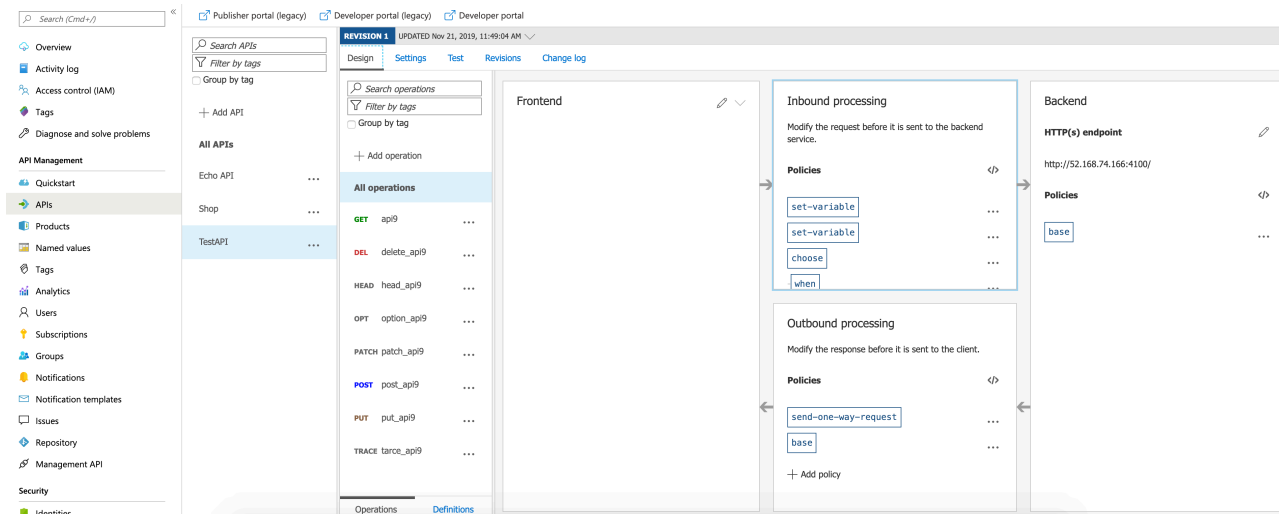
If you change any of the **Named Values** after the policy is operational, it takes 60-seconds for the change to be applicable. For example, if you change the ase-primary node IP address, the new IP address would take effect only after 60-seconds.

 **Note:** Make sure that the ASE primary and secondary IP address is followed by a /

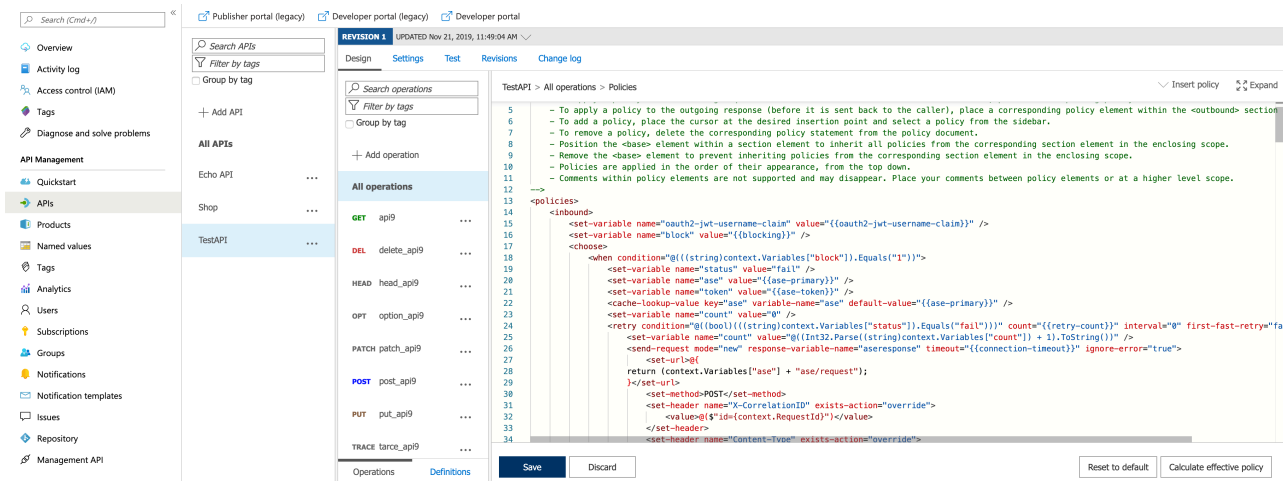
Display name	Value	Tags
ase-primary	http://<primary_ase_address>:<port>/	...
ase-secondary	http://<secondary_ase_address>:<port>/	...
ase-token	your_ase_sideband_token	...
blocking	1	...
connection-timeout	20	...
oauth2-jwt-username-claim	username	...
retry-count	2	...

- 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.



- Click on the **Save** button to save the policy



**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.

## 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 15
- [API naming guidelines](#) on page 50
- Adding APIs in [Sideband ASE](#) on page 42. You can add individual APIs or you can configure a global API. For more information, see [API discovery](#) on page 203.

- [Configure ASE to ABS connectivity](#) on page 66

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 195
- [API reports using Postman](#) on page 279
- [Access the PingIntelligence Dashboard](#) on page 373

## 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

APIs can be automatically discovered using the PingIntelligence ABS AI Engine. For more information on enabling discovery. For more information on enabling discovery, see [Enable and disable discovery](#). APIs are discovered when a global API JSON file is added to PingIntelligence ASE. For more information, see [API discovery](#). After the APIs are discovered by ABS, AAD adds the API JSON to ASE. Install AAD to add discovered APIs to ASE.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD logs
- `util` - Contains the `check_ports.sh`. Run on the machine with the AAD tool to check if ASE and ABS default ports are open.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	NA
<code>aad.env.dev.fullsync</code>	NA
<code>aad.mode</code>	Set the value to <code>discovery</code> for AAD to work in discovery mode.
<code>abs.host</code>	ABS host IP address
<code>abs.port</code>	ABS host port number

<code>abs.access_key</code>	ABS access key
<code>abs.secret_key</code>	ABS secret key
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	Set to true for ABS-AAD communication to use SSL.
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>
<code>abs.query.interval</code>	NA.
<code>aad.log.level</code>	The log level of AAD log files. The default value is <code>INFO</code> . Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code>
<code>gateway.management.url</code>	NA
<code>gateway.management.username</code>	NA
<code>gateway.management.password</code>	NA
<code>pingaccess.management.url</code>	NA
<code>pingaccess.management.username</code>	NA
<code>pingaccess.management.password</code>	NA

Following is a sample `aad.properties` file:

```
# Automated API Discovery (AAD)

# Automated API Discovery (AAD)

# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod

# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
# or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
```

```

# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMaOMtLL8hUPOLWrq7E
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMaOMtLL8hUPOLWrq7E

### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsjTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSmv5mZllw3WpAsj1oPF3d5Et170Y=
# ABS secret key
abs.secret_key=OBF:AES:RsjTC/tx/sp
+7XXtr8+lrnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74LahxfEgp:dLfcNugFUCcfsq1nBXQzflTvAWiPit8ulseHxi+Z0tk=

### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+lpEqcKQnFG/UN3Efz0DMa/
kmI=:Az82rlUFftMGpMxF7unelJZUucX191102QgKvHD36vU=

```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `gateway.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

## Generate aad\_master.key

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```
opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

## Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start AAD after passwords are obfuscated.

## Start AAD

### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

## Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/stop.sh
```

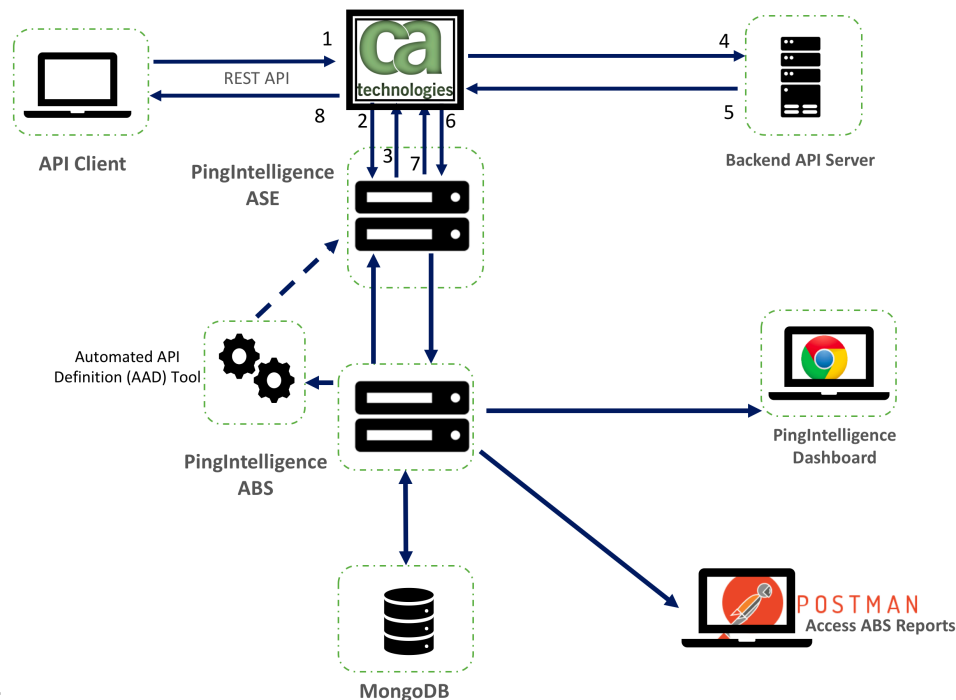
Ping Identity Inc.  
AAD is stopped.

## 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. 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: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: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: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: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 537.

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):

```

17  > Comment: *
18  [i] ASE Check Request // create an ASE validation request for the
19  [i] Comment: *
20  [i] Comment: * Example for 'continue processing'
21  ▶ [x] At least one assertion must evaluate to true // Audit a message
24  [i] Comment: *
25  [g] Route via HTTPS to https://localhost:8443/shop/backend // si
26  [i] ASE Check Response // create an ASE validation request for the

```

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.

## 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 15
- [API naming guidelines](#) on page 50
- Adding APIs in [Sideband ASE](#) on page 42. You can add individual APIs or you can configure a global API. For more information, see [API discovery](#) on page 203.
- [Configure ASE to ABS connectivity](#) on page 66

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 195
- [API reports using Postman](#) on page 279
- [Access the PingIntelligence Dashboard](#) on page 373

## API discovery

Automatically build definitions for your API gateway in your environment by using PingIntelligence ABS AI Engine. For more information on enabling automatic discovery, see [Enable and disable discovery](#). APIs are discovered when a global API JSON is defined in the ASE. For more information, see [API discovery](#). After the APIs are discovered by the ABS AI Engine, AAD adds the API JSON file to ASE. Install AAD to add discovered API definitions to ASE.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11.0.2 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD's logs



- `util` - Contains the `check_ports.sh`. Run on the machine with the AAD tool to check ASE and ABS default ports.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	NA
<code>aad.env.dev.fullsync</code>	NA
<code>aad.mode</code>	Set the value to <code>discovery</code> for AAD to work in discovery mode.
<code>abs.host</code>	ABS host IP address
<code>abs.port</code>	ABS host port number
<code>abs.access_key</code>	ABS access key
<code>abs.secret_key</code>	ABS secret key
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	Set to true for ABS-AAD communication to use SSL.
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>
<code>abs.query.interval</code>	NA.
<code>aad.log.level</code>	The log level of AAD log files. The default value is <code>INFO</code> . Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code>
<code>gateway.management.url</code>	NA
<code>gateway.management.username</code>	NA
<code>gateway.management.password</code>	NA
<code>pingaccess.management.url</code>	NA
<code>pingaccess.management.username</code>	NA
<code>pingaccess.management.password</code>	NA

Following is a sample `aad.properties` file:

```
# Automated API Discovery (AAD)
# Automated API Discovery (AAD)
# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod
# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
```

```

# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E

### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsjTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSmv5mZllw3WpAsj1oPF3d5Et170Y=
# ABS secret key
abs.secret_key=OBF:AES:RsjTC/tx/sp
+7XXtr8+1rnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEaojYV/
Otw74LahxfEgp:dLfcNugFUCcfsqlnBXQzflTvwAWiPit8ulseHxi+Z0tk=

### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+lpEqcKQnFG/UN3Efz0DMA/
kmI=:Az82rlUfftMGpMxF7unelJZUucX191102QgKvHD36vU=

```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- ase.access\_key
- ase.secret\_key
- gateway.management.password

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

### Generate `aad_master.key`

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```
opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start AAD after passwords are obfuscated.

### Start AAD

#### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

### Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/stop.sh
Ping Identity Inc.
AAD is stopped.
```

## PingAccess API gateway integration

---

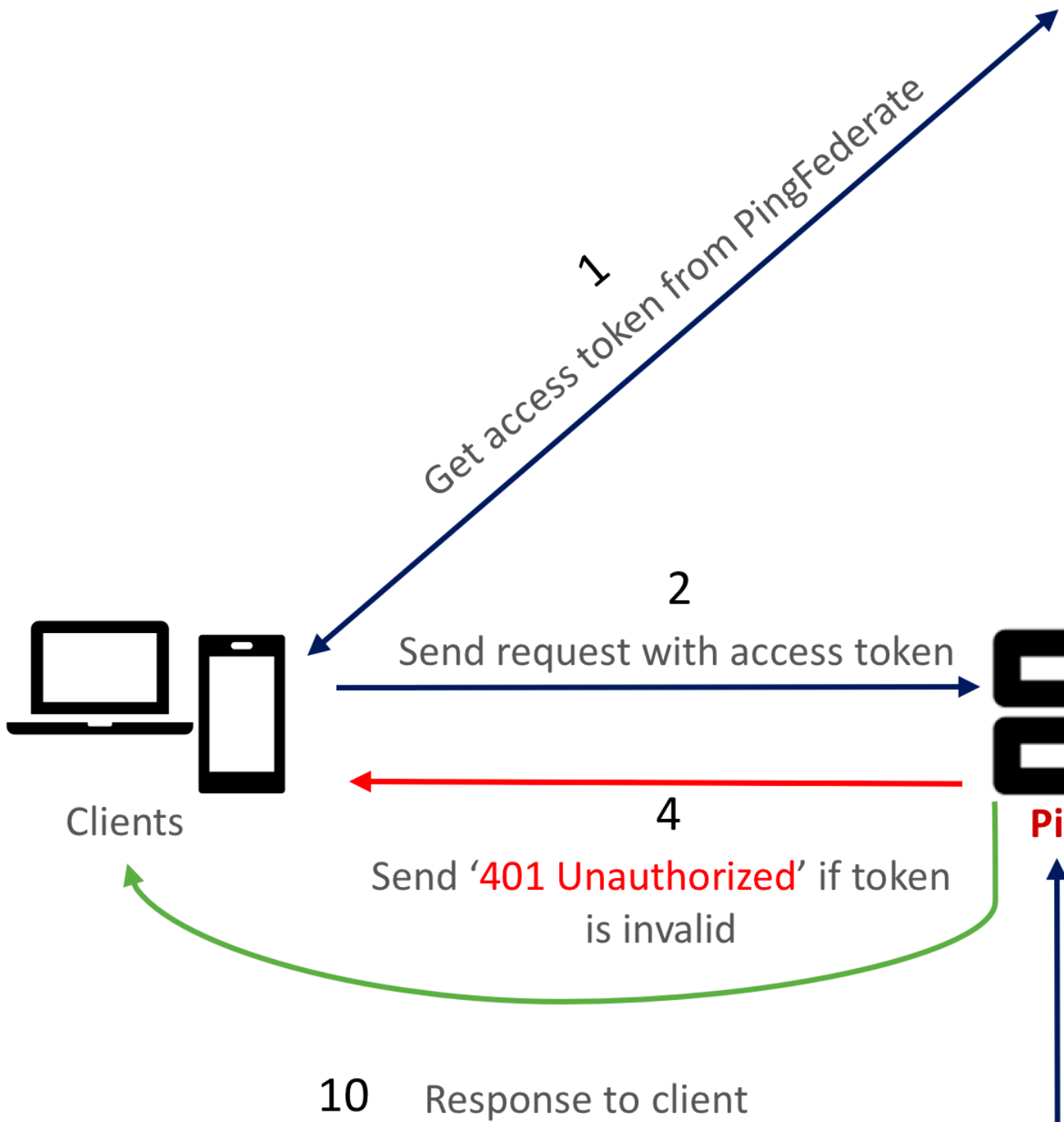
### PingAccess sideband integration

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. PingIntelligence 4.0 software adds support for reporting and attack detection based on usernames captured from token attributes.

PingIntelligence provides the following three policies based on the version of OpenJDK or Oracle JDK:

- **OpenJDK 11:** `pingIntelligenceASE-openjdk_11`
- **Oracle JDK 8:** `pingIntelligenceASE-oraclejdk_8`
- **Oracle JDK 11:** `pingIntelligenceASE-oraclejdk_11`

This 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 origin IP, cookie or OAuth2 token against the AI generated Blacklist. If all checks pass, ASE returns a 200-OK response to the PingAccess. If not, a different response code is sent to PingAccess. The request information is also logged by ASE and sent to the AI Engine for processing.
7. If PingAccess 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 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 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** - PingIntelligence 4.0 works with PingAccess 5.2.
- **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 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                   : 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 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.

- **Port for AAD**

If you are using AAD to automate API definition updates on PingIntelligence, open the following ports:

- Open the AAD management port to fetch API definitions from PingAccess. The default port is 9000.
- Open port 8010 in ASE for AAD to add API definitions.

### 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

The screenshot shows the PingFederate web interface. The left sidebar has a navigation menu with 'MAIN' (Identity Provider, OAuth Server) and 'SETTINGS' (Security, System). The main content area is titled 'Access Token Management | Create Access Token Management Instance'. There are four tabs: 'Type', 'Instance Configuration', 'Session Validation', and 'Access Token Attribute Contract' (which is active). Below the tabs, there is a text prompt: 'Provide the names of the attributes that will be carried in (or referenced by) the OAuth access token. For auditing purposes, you must include the attribute name.' Underneath, there is a section 'Extend the Contract' with a list of attributes: 'email', 'group', 'org', and 'Username'. The 'Username' attribute is highlighted, and a text input field is shown below it. Below the text input, there is a 'Subject Attribute Name' dropdown menu with 'USER\_KEY' selected.

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**

The screenshot shows the PingFederate web interface. The left sidebar has a 'MAIN' section with 'Identity Provider' and 'OAuth Server' (selected), and a 'SETTINGS' section with 'Security' and 'System'. The main content area is titled 'Access Token Attribute Mapping | Access Token Mapping' and has four tabs: 'Attribute Sources & User Lookup', 'Contract Fulfilment', 'Issuance Criteria', and 'Summary'. The 'Contract Fulfilment' tab is active. Below the tabs, there is a instruction: 'Select a Source and Value to map into each item in the Contract list.' A table with two columns, 'Contract' and 'Source', is shown. The rows are: 'Username' with 'Persistent Grant', 'email' with 'Text', 'group' with 'Text', and 'org' with 'Context'.

## Deploy the PingIntelligence policy

About this task

To integrate PingAccess with PingIntelligence components, complete the following steps on PingAccess:

Steps

1. Download the PingAccess Policy from the [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.
2. Change the name of the \*.jar file to PingIntelligence.jar.
3. Copy the PingIntelligence.jar file into the lib directory in PA\_home.
4. Restart PingAccess
5. Add Applications in PingAccess with **Application Type** as API. For AAD to automatically capture the definition, include the following parameters in the **DESCRIPTION** section when you add an Application:

```
{
  "ping_ai": true,
  "ping_host": "",
}
```



```

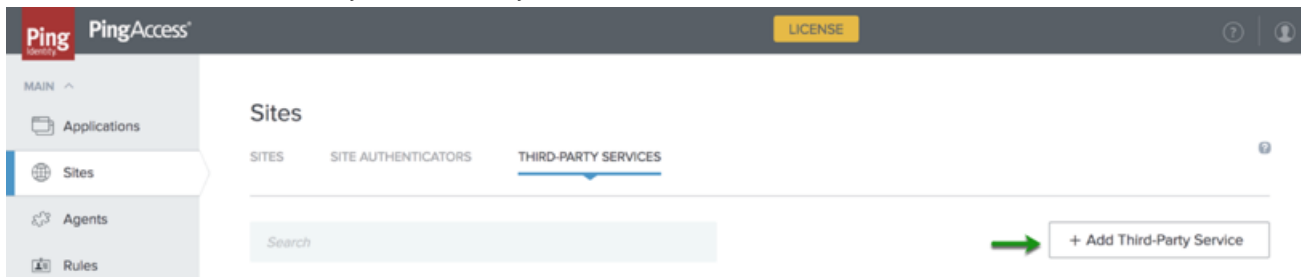
"ping_url": "",
"ping_login": "",
"ping_cookie": "JSESSIONIDTEST",
"apikey_qs": "X-API-KEY",
"apikey_header": "",
"ping_decoy": false,
"oauth2_access_token": false,
"ping_blocking": true
}

```

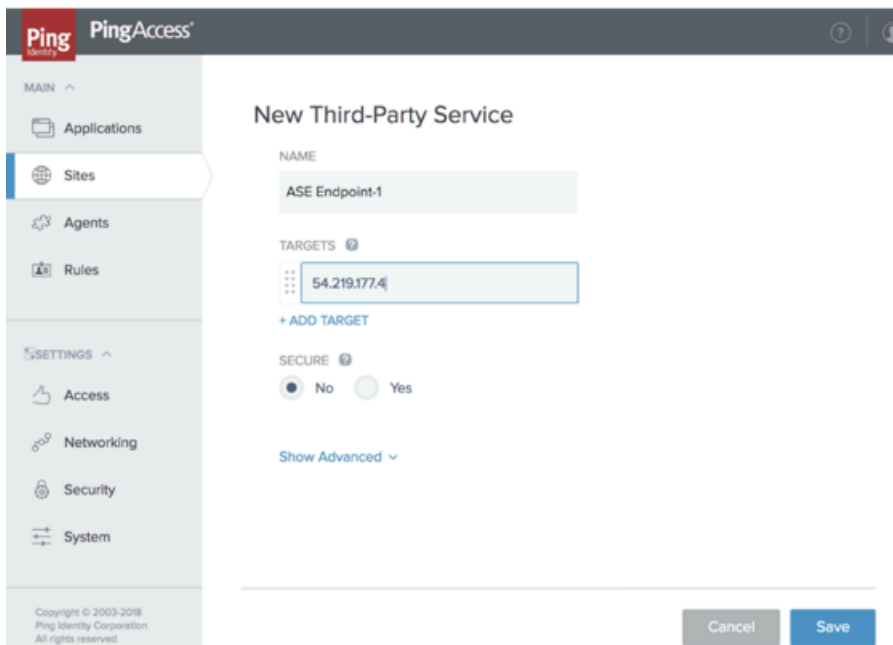
The following table describes the parameters captured when [AAD](#) fetches the API definition from PingAccess and adds it to ASE:

Parameter	Description
ping_ai	When <code>true</code> , PingIntelligence processing is applied to this API. Set to <code>false</code> for no PingIntelligence processing. Default value is <code>true</code> .
ping_host	Hostname of the API. You can configure <code>*</code> as <code>hostname</code> to support any hostname.
ping_url	The base URL of the managed API, for example, <code>/shopping</code> . 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 <code>true</code> , API is a decoy API. The values can be <code>true</code> or <code>false</code> .
oauth2_access_token	When <code>true</code> , PingIntelligence expects an OAuth token. The values can be <code>true</code> or <code>false</code> .
ping_blocking	When <code>true</code> , enable PingIntelligence blocking when attack are detected on the API. The default value is <code>true</code> . To disable blocking for the API, set to <code>false</code> .

6. Add two ASEs for redundancy in Third-Party Services.



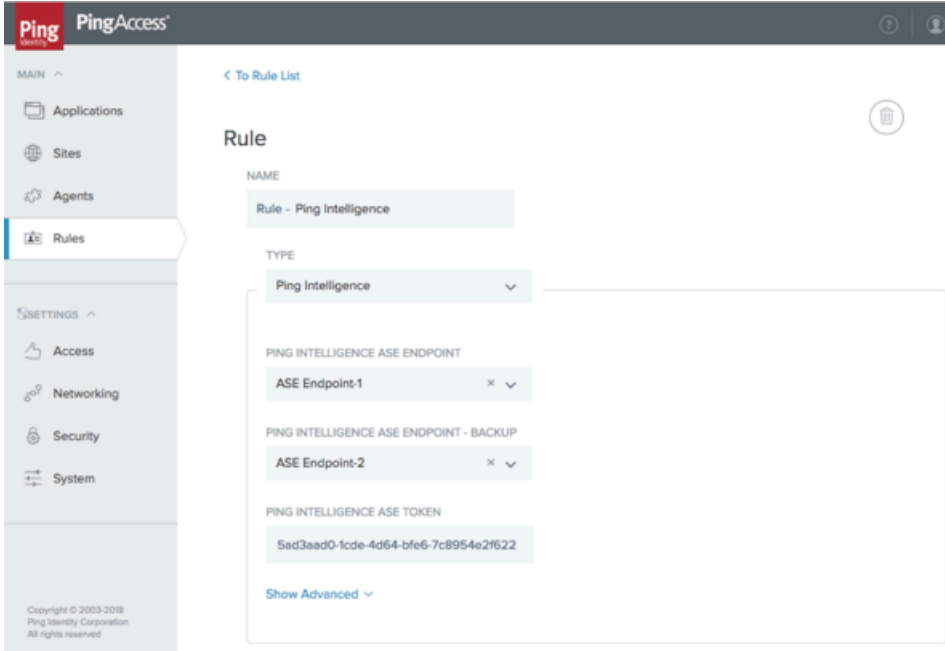
In the New Third-Party Service page, add two ASE IP addresses.



7. Add a Rule for the two ASEs. Click **Rules** Tab. In the New Rule page, enter the name of the rule for PingIntelligence. In the **TYPE** drop-down list, select **Ping Intelligence**. This appears in the drop-down list after adding the PingIntelligence.jar in PA\_Home in step 1.

8. Select the **ASE SERVICE ENDPOINT** to which you want to apply the rule.

9. Add the generated ASE sideband token which is used for authentication between PingAccess and



The screenshot shows the PingAccess web interface. The left sidebar contains navigation menus for 'MAIN' (Applications, Sites, Agents, Rules) and 'SETTINGS' (Access, Networking, Security, System). The main content area is titled 'Rule' and shows the configuration for a rule named 'Rule - Ping Intelligence'. The configuration includes:

- NAME: Rule - Ping Intelligence
- TYPE: Ping Intelligence
- PING INTELLIGENCE ASE ENDPOINT: ASE Endpoint-1
- PING INTELLIGENCE ASE ENDPOINT - BACKUP: ASE Endpoint-2
- PING INTELLIGENCE ASE TOKEN: 5ed3aad0-1cde-4d64-bfe6-7c8954e2f622

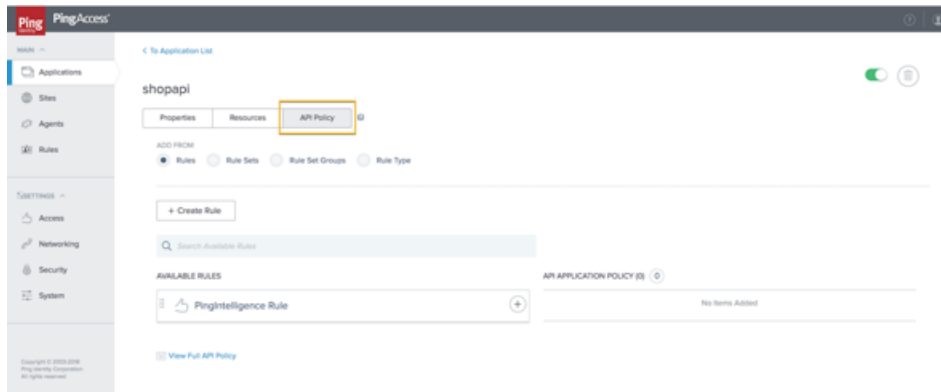
A 'Show Advanced' link is visible at the bottom of the configuration area.

ASE.

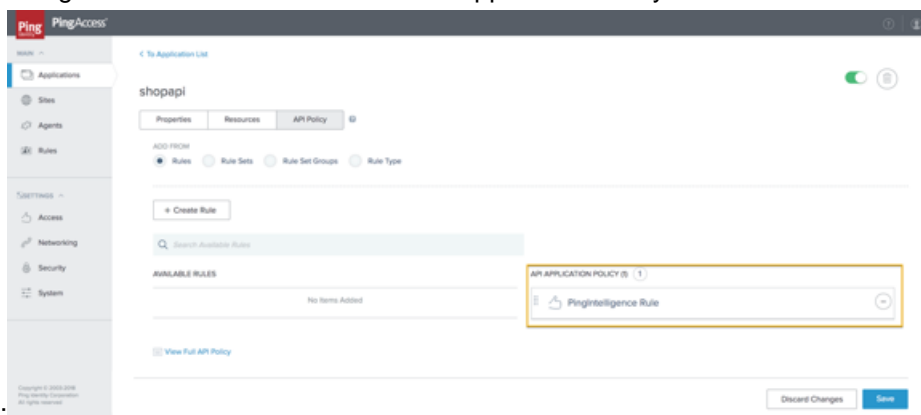
**Note:** In the event of an ASE node failure, traffic is automatically routed to the standby ASE node.

10. Apply the rule to the application 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  $\oplus$  sign for the PingIntelligence rule. After clicking on the sign, the PingIntelligence Rule moves under the API Application Policy as shown in the screen



capture below:

- d. Save the rule by clicking on the **Save** button.

**Note:** It is a good practice to connect PingAccess to PingIntelligence ASE using HTTPS.

### 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

Install and configure AAD to automatically capture API definitions from PingAccess. AAD discovers these API definitions, converts them into PingIntelligence API JSON files, and adds the definitions to ASE. For detailed information on AAD, see Automated API Definition tool information in the [ABS admin guide](#).

AAD supports two operating modes:

- Dev – PingAccess API definitions are mirrored in the PingIntelligence environment. If an API is removed from PingAccess, it is deleted from PingIntelligence.
- Prod - PingAccess API definitions are mirrored in the PingIntelligence environment but APIs are not deleted from PingIntelligence.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11 must already be installed on the AAD machine.


Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD's logs
- `util` - Contains `thecheck_ports.sh`. Run on the machine with the AAD tool to check ASE and ABS default ports.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	<p>Set the key to <code>dev</code> if you are fetching API definitions in an API development environment.</p> <p>Set the key to <code>prod</code> if you are fetching API definitions in a production environment</p>
<code>aad.env.dev.fullsync</code>	<p>Set to <code>true</code> to mirror the PingAccess API definitions in PingIntelligence. AAD deletes any APIs from PingIntelligence that are removed from PingAccess.</p> <p>Set to <code>false</code> to synchronize PingAccess and PingIntelligence definitions without deleting other APIs present in PingIntelligence.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> <b>Note:</b> <code>aad.env.dev.fullsync</code> key is only valid in a development environment (<code>aad.env=dev</code>)</p> </div>
<code>aad.mode</code>	Set the value to <code>pingaccess</code> when ASE is deployed in sideband mode with PingAccess
<code>abs.host</code>	NA
<code>abs.port</code>	NA
<code>abs.access_key</code>	NA
<code>abs.secret_key</code>	NA
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	NA
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>

<code>abs.query.interval</code>	NA.
<code>aad.log.level</code>	The log level of AAD log files. The default value is INFO. Other possible values are: ALL<DEBUG<INFO<WARN<ERROR<FATAL<OFF
<code>gateway.management.url</code>	NA
<code>gateway.management.username</code>	NA
<code>gateway.management.password</code>	NA
<code>pingaccess.management.url</code>	PingAccess management URL Only valid when <code>aad.mode</code> is <code>pingaccess</code> .
<code>pingaccess.management.username</code>	PingAccess management Username Only valid when <code>aad.mode</code> is <code>pingaccess</code> .
<code>pingaccess.management.password</code>	PingAccess management Password. Only valid when <code>aad.mode</code> is <code>pingaccess</code> .

Following is a sample `aad.properties` file:

```
# Automated API Discovery (AAD)
# Automated API Discovery (AAD)

# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod

# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
# or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
```

```

# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7

### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsjTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSmv5mZllw3WpAsjl0PF3d5Et170Y=
# ABS secret key
abs.secret_key=OBF:AES:RsjTC/tx/sp
+7XXtr8+1rnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74LahxfEgp:dLfcNugFUCcfsglnBXQzflTvwAWiPit8ulseHxi+Z0tk=

### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+lpEqcKQnFG/UN3Ezfz0DMA/
kmI=:Az82rlUFftMGpmx7F7unelJZUucX191102QgKvHD36vU=


```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `gateway.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

### Generate `aad_master.key`

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```

opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys

```

```
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
  config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
  abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start AAD after passwords are obfuscated.

### Start AAD

#### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

### Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

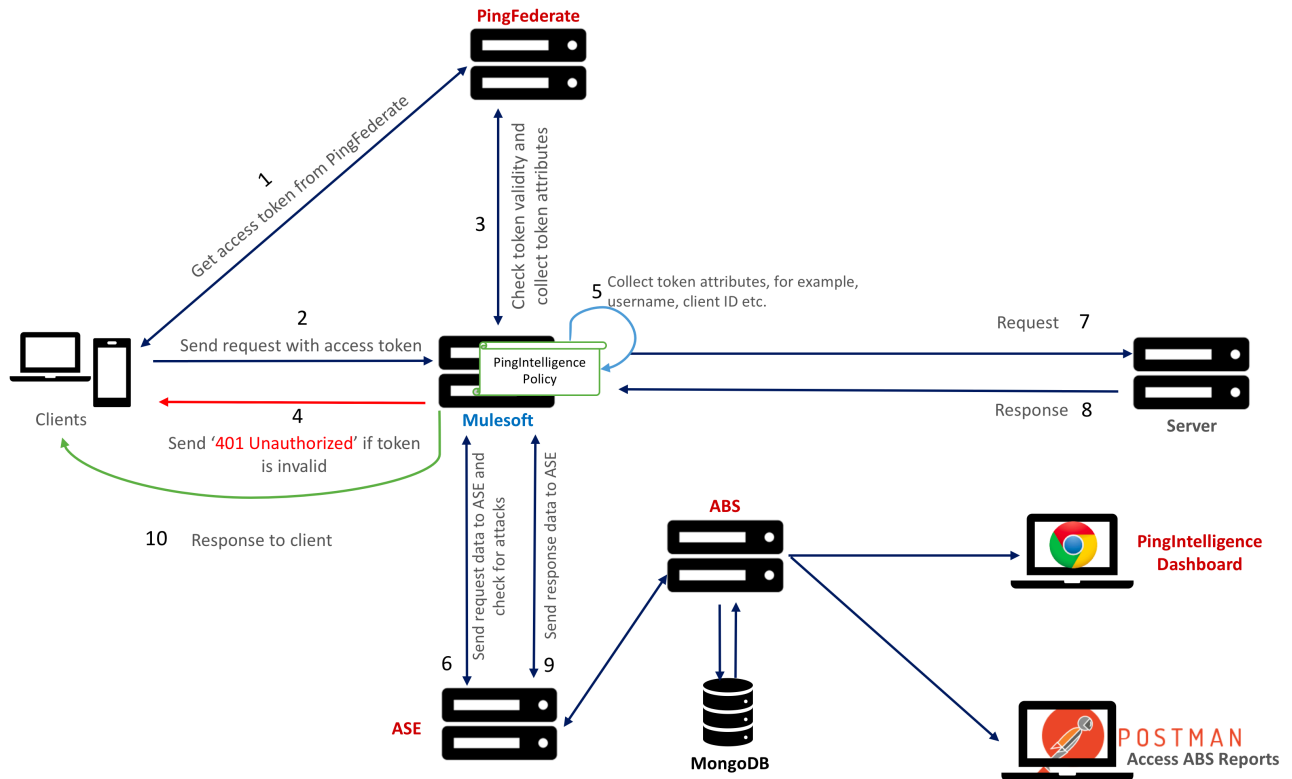
```
bin/stop.sh
Ping Identity Inc.
AAD is stopped.
```



## 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.



Here is the traffic flow through the Mulesoft 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. Mulesoft verifies the authenticity of the access token with PingFederate.
4. If the token is invalid, Mulesoft returns a 401-unauthorized message to the client.
5. If the token is valid, the PingIntelligence policy running in Mulesoft collects API metadata and token attributes.
6. Mulesoft makes an API call to send the request information to ASE. ASE checks the request against a registered set of APIs and checks the origin IP, cookie or OAuth2 token against the 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:

- **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             : 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 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:

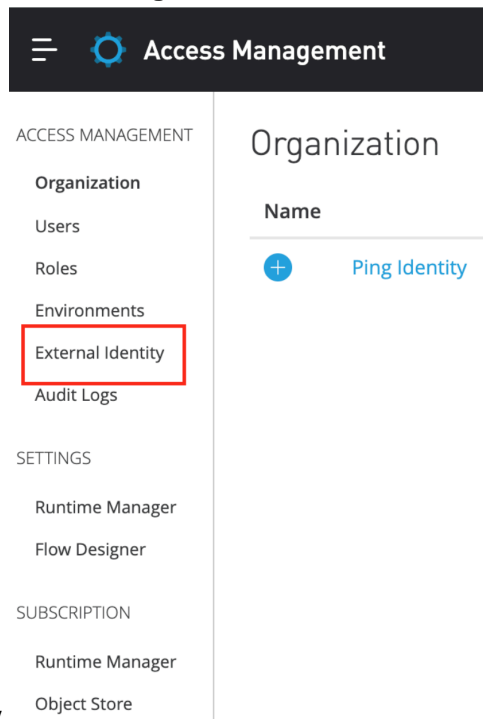
```
# ./bin/cli.sh -u admin -p admin create_sideband_token
```

Save the generated authentication token for further use.

**Create an External Identity in Mulesoft:** Configure an **External Identity** to integrate PingFederate with Mulesoft. For enabling this, log in to your organization's Mulesoft Any point platform. Complete the following steps:

1. Log in to your organization's Mulesoft Any Point and click on **Access Management**.

2. In the **Access Management** window, click on **External**



**Identity.**

External Identity page is displayed.

3. In the External Identity page, under **Client Management**, select **PingFederate**. Enter the details. Fetch the **Client ID** and **Client Secret** information from PingFederate. Client ID and Client Secret are the credentials used to generate tokens.

4.

## 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.

The screenshot shows the API Manager interface for the 'userinfo-mule39 v1' API. The API is active and has an asset version of 1.0.0. A 'Configuration Detail' modal is open, showing the following settings for the 'PingFederate access token enforcement' policy:

Property	Value
Policy	PingFederate access token enforcement
Scopes	openid profile email
Skip Client Id Validation	true

Below the modal, the 'API level policies' section shows a table of applied policies:

Name	Category	Fulfills						
<table border="1"> <thead> <tr> <th>Order</th> <th>Method</th> <th>Resource URI</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>All API Methods</td> <td>All API Resources</td> </tr> </tbody> </table>	Order	Method	Resource URI	1	All API Methods	All API Resources	PingFederate access token enforcement	OAuth 2.0 protected
Order	Method	Resource URI						
1	All API Methods	All API Resources						
PingIntelligence	Security	Message protection						

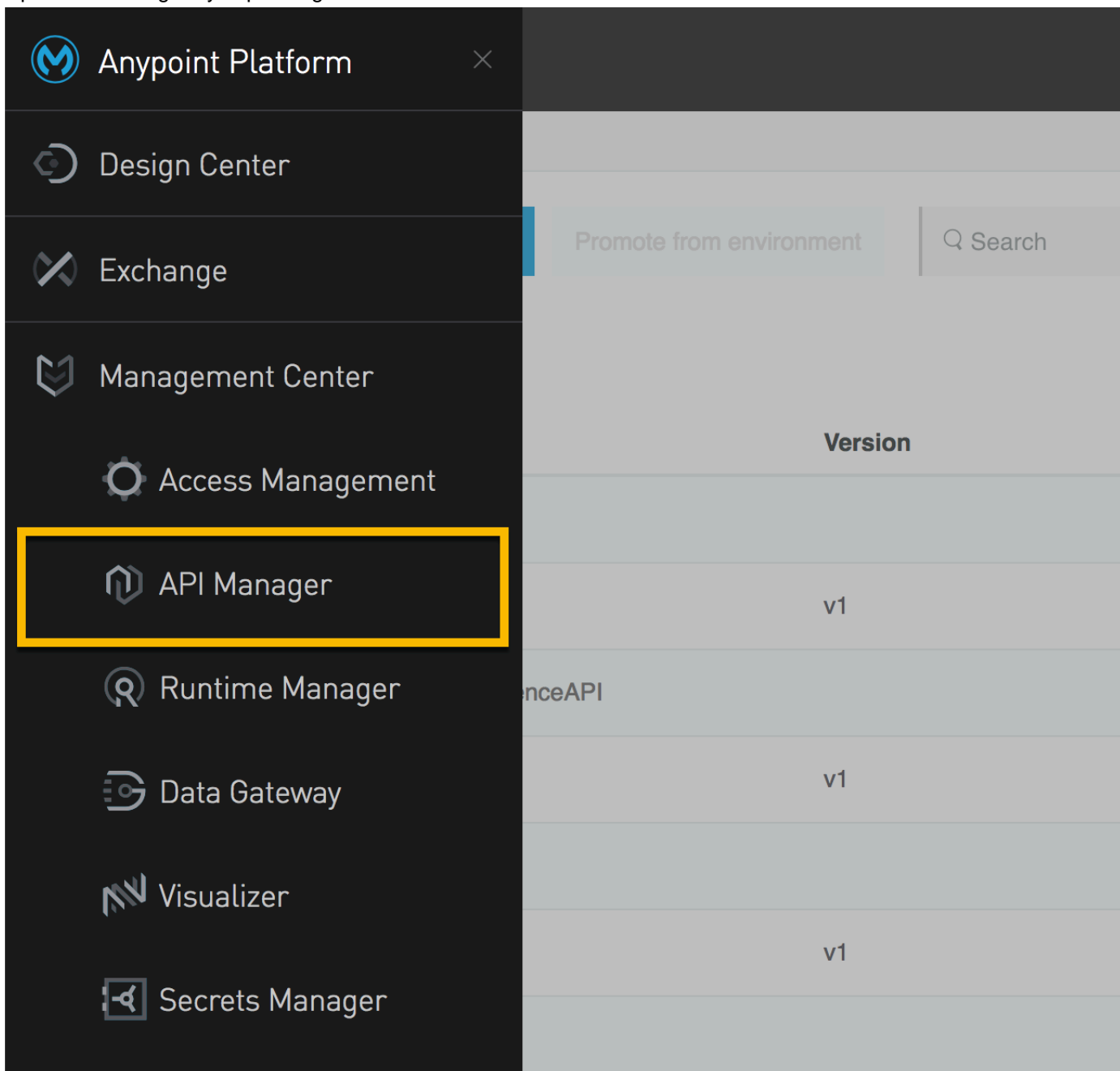
### PingIntelligence for Mulesoft 3.9

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.

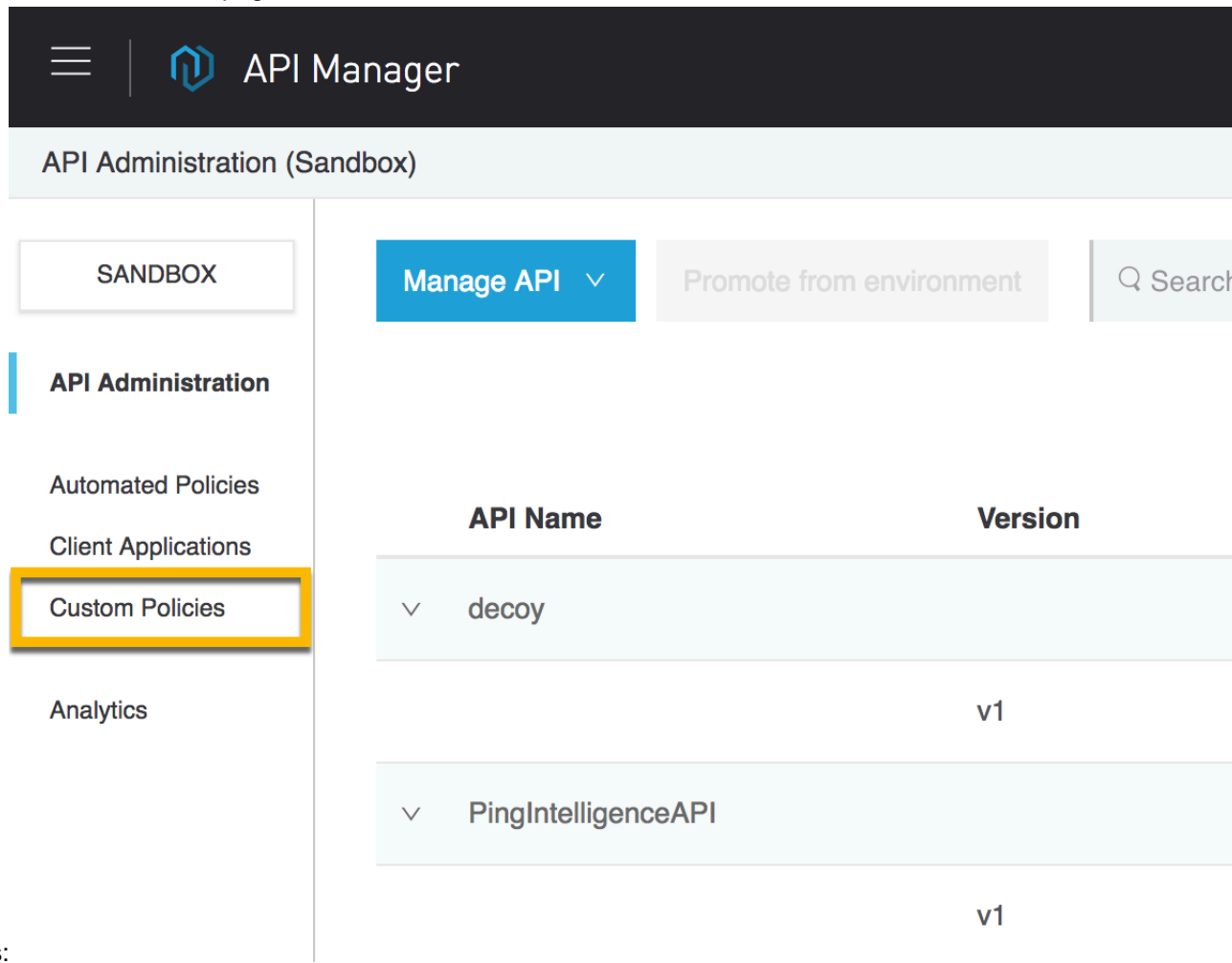
#### Deploying PingIntelligence policy to Mulesoft Anypoint

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



The screenshot shows the API Manager interface. At the top, there is a dark header with a hamburger menu icon, the API Manager logo, and the text "API Manager". Below the header, the page title is "API Administration (Sandbox)".

On the left side, there is a navigation menu with the following items:

- SANDBOX
- API Administration** (highlighted with a blue bar)
- Automated Policies
- Client Applications
- Custom Policies** (highlighted with a yellow box)
- Analytics

At the top of the main content area, there are three buttons: "Manage API" (blue), "Promote from environment" (grey), and a search bar with a magnifying glass icon and the text "Search".

The main content area displays a table with the following columns: "API Name" and "Version".

API Name	Version
▼ decoy	v1
▼ PingIntelligenceAPI	v1

Below the table, the text "Policies:" is visible.

4. In the **Custom policies** page, click on **Add custom**

API Manager

Custom Policies

SANDBOX

API Administration

Automated Policies

Client Applications

**Custom Policies**

policy: Analytics

## Custom policies

Create **custom policies** that will be available to all APIs in your orga

The policies listed on this page were created for **Mule 3.9, and earl**

In Mule 4 and later, policies are stored as assets in Exchange. Ther

**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`

Add custom policy ×

**Mule Version**

Policy for runtimes older than Mule 4  Policy for Mule 4 or later

**Name \***

|

PingIntelligence Policy

A YAML file defining the configuration parameters for the policy.

No file chosen

**Policy configuration \***

The Mule configuration XML for the policy implementation. Learn more [here](#).

No file chosen

files: \_\_\_\_\_



PingIntelligence Policy is added as shown below:

The screenshot shows the API Manager interface. The top navigation bar includes a menu icon and the 'API Manager' logo. Below this is a 'Custom Policies' header. On the left, a sidebar lists navigation options: 'SANDBOX', 'API Administration', 'Automated Policies', 'Client Applications', 'Custom Policies' (which is highlighted with a blue bar), and 'Analytics'. The main content area is titled 'Custom policies' and contains the text: 'Create custom policies that will be available for your APIs. The policies listed on this page were created in Mule 3. The policies listed on this page were created in Mule 4 and later, policies are stored in the API Manager.' Below this text is a blue button labeled 'Add custom policy'. Underneath the button, a table with a single row is visible, showing the name 'PingIntelligence Policy'.

#### PingIntelligence for Mulesoft 4.0

Before applying the PingIntelligence policy, make sure that the API to which you want to apply the policy is defined. To deploy the PingIntelligence policy for Mulesoft 4.0, upload the policy to Exchange.

Follow the steps mentioned in [Getting started with Custom Policies development](#) link to upload the PingIntelligence policy.

When the project's directory structure is created, replace the contents of `my-custom-policy.yaml` with that of `pi_policy.yaml` file. Similarly, replace the contents of `template.xml` with that of `pi_policy.xml`. The following screen shot shows the reference directory structure created by following the steps mentioned at the [Getting started with Custom Policies development](#):

```

my-custom-policy/
├── my-custom-policy.yaml ← Copy the contents of pi_policy.yaml
├── mule-artifact.json
├── pom.xml
├── src
│   ├── main
│   │   └── mule
│   │       └── template.xml ← Copy the contents of pi_policy.xml

```

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>

```

After the project is setup, complete the steps in the packaging and uploading links for the PingIntelligence policy:

- [Packaging a Custom Policy](#)
- [Uploading a Custom Policy to Exchange](#)

## Apply PingIntelligence policy

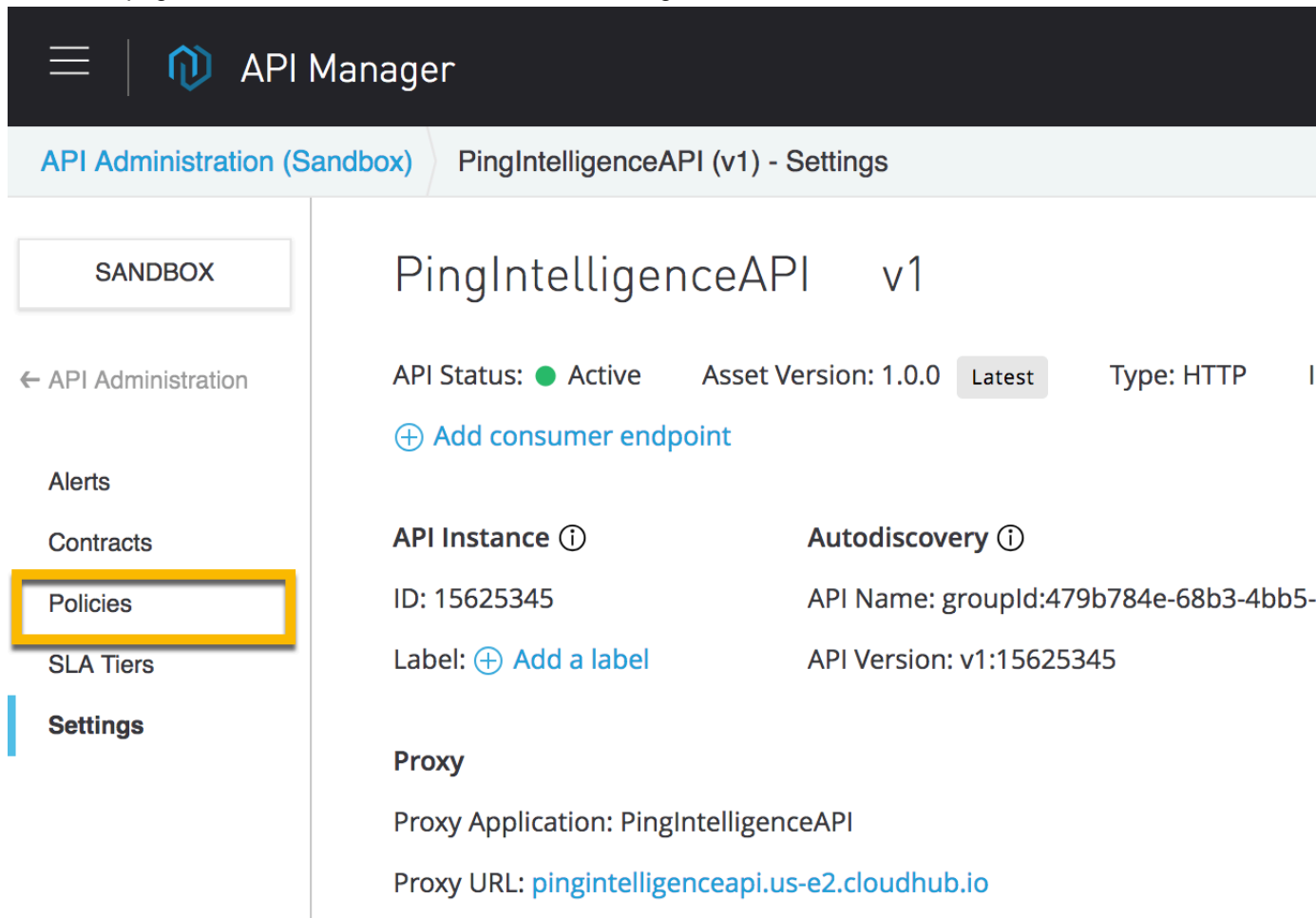
Complete the following steps to attach the PingIntelligence policy to your API:

1. Navigate to the API manager and click on the **Version** of the API to which you want to attach the PingIntelligence policy:

The screenshot shows the API Manager interface. At the top, there is a navigation bar with a hamburger menu icon, the API Manager logo, and the text 'API Manager'. Below this is a header for 'API Administration (Sandbox)'. On the left side, there is a sidebar with a 'SANDBOX' button and a list of navigation items: 'API Administration', 'Automated Policies', 'Client Applications', 'Custom Policies', and 'Analytics'. The main content area features a 'Manage API' button (highlighted in blue) and a 'Promote from environment' button (disabled). Below these buttons is a table with two columns: 'API Name' and 'Version'. The table contains two rows: one for 'decoy' with version 'v1', and one for 'PingIntelligenceAPI' with version 'v1'. The 'PingIntelligenceAPI' row is highlighted with a yellow border.

API Name	Version
decoy	v1
PingIntelligenceAPI	v1

2. In the API page, click on **Policies** as shown in the following illustration:



The screenshot displays the API Manager interface. The top navigation bar includes a hamburger menu icon and the text 'API Manager'. Below this, a breadcrumb trail shows 'API Administration (Sandbox)' and 'PingIntelligenceAPI (v1) - Settings'. The left sidebar contains a 'SANDBOX' button and a list of navigation items: 'API Administration', 'Alerts', 'Contracts', 'Policies' (highlighted with a yellow border), 'SLA Tiers', and 'Settings'. The main content area shows the details for 'PingIntelligenceAPI v1', including its status (Active), asset version (1.0.0), and type (HTTP). It also features a link to 'Add consumer endpoint' and two sections: 'API Instance' and 'Autodiscovery'. The 'API Instance' section lists the ID (15625345) and a label link. The 'Autodiscovery' section lists the API Name and API Version. A 'Proxy' section at the bottom shows the application name and the proxy URL.

API Manager

API Administration (Sandbox) PingIntelligenceAPI (v1) - Settings

SANDBOX

← API Administration

Alerts

Contracts

**Policies**

SLA Tiers

Settings

## PingIntelligenceAPI v1

API Status: ● Active    Asset Version: 1.0.0 **Latest**    Type: HTTP

[+ Add consumer endpoint](#)

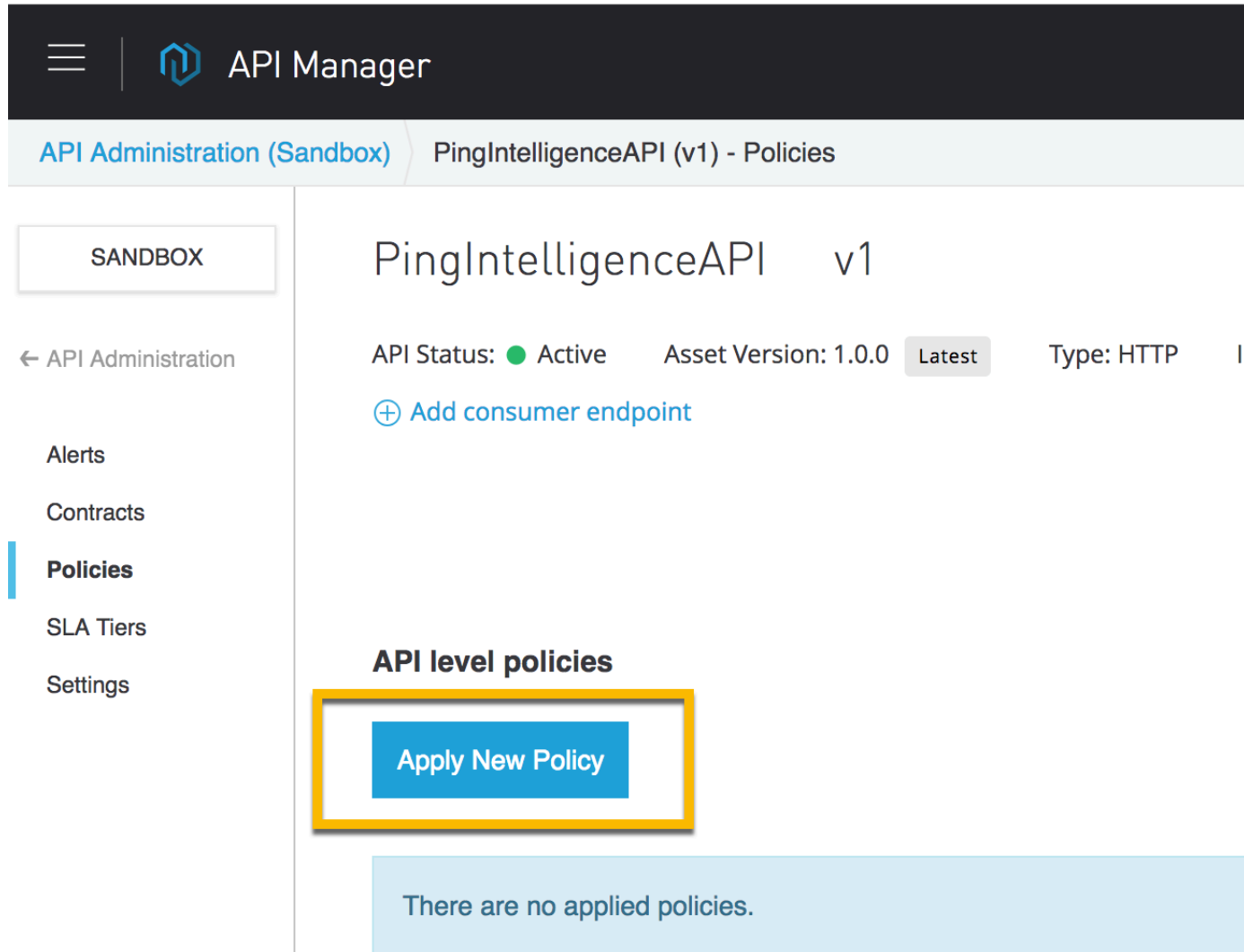
<b>API Instance</b> ⓘ	<b>Autodiscovery</b> ⓘ
ID: 15625345	API Name: groupId:479b784e-68b3-4bb5-
Label: <a href="#">+ Add a label</a>	API Version: v1:15625345

**Proxy**

Proxy Application: PingIntelligenceAPI

Proxy URL: [pingintelligenceapi.us-e2.cloudhub.io](https://pingintelligenceapi.us-e2.cloudhub.io)

The Policies page supports applying the PingIntelligence policy to the API. Click on **Apply New**:



API Manager

API Administration (Sandbox) PingIntelligenceAPI (v1) - Policies

SANDBOX

← API Administration

Alerts

Contracts

**Policies**

SLA Tiers

Settings

PingIntelligenceAPI v1

API Status: ● Active Asset Version: 1.0.0 Latest Type: HTTP

⊕ Add consumer endpoint

**API level policies**

Apply New Policy

There are no applied policies.

3. In the **Select Policy** pop-up window, select the PingIntelligence Policy and click on **Configure**

## Select Policy

All Categories ∨ Fullfills

Policies	Requ
<input type="radio"/> Client ID enforcement ⓘ	—
<input type="radio"/> HTTP basic authentication ⓘ	Secur
<input type="radio"/> IP blacklist ⓘ	—
<input type="radio"/> IP whitelist ⓘ	—
<input type="radio"/> JSON threat protection ⓘ	—
<input type="radio"/> LDAP security manager ⓘ	—
<input type="radio"/> OAuth 2.0 access token enforcement using external provider ⓘ	—
<input checked="" type="radio"/> PingIntelligence Policy <span>Custom</span> ⓘ	—

**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).

## Apply PingIntelligence Policy policy

ASE sideband Policy by PING for Mule 3.9.X APIs deployed on Mule Cloudhub

### ASE TOKEN

ASE sideband authentication token

a426203606db4a00a0e4e4207ad603fe

### ASE Primary Host \*

ASE HOSTNAME:PORT

192.168.11.28:80

### ASE Secondary Host \*

ASE HOSTNAME:PORT for failover

192.168.11.30:80

Enable SSL

If enabled, Mulesoft will connect to ASE over HTTPS

Allow self-signed certificate

If enabled, Mulesoft will accept self-signed certificate from ASE

### Connection Timeout \*

Connection timeout in milliseconds

5000

### Read Timeout \*

Read timeout in milliseconds

5000

Cancel

Apply

**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 6. In the API page, scroll down to the **Deployment Configuration** section and click on

## Deployment Configuration ▼

Runtime version:

β.9.x

Proxy application name: ⓘ

PingIntelligenceAPI

**Redeploy:**

### 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](#) on page 203.
- [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.](#)

### API discovery

Automatically build definitions for your API gateway in your environment by using PingIntelligence ABS AI Engine. For more information on enabling automatic discovery, see [Enable and disable discovery](#). APIs are discovered when a global API JSON is defined in the ASE. For more information, see [API discovery](#). After



the APIs are discovered by the ABS AI Engine, AAD adds the API JSON file to ASE. Install AAD to add discovered API definitions to ASE.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11.0.2 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD's logs
- `util` - Contains the `check_ports.sh`. Run on the machine with the AAD tool to check ASE and ABS default ports.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	NA
<code>aad.env.dev.fullsync</code>	NA
<code>aad.mode</code>	Set the value to <code>discovery</code> for AAD to work in discovery mode.
<code>abs.host</code>	ABS host IP address
<code>abs.port</code>	ABS host port number
<code>abs.access_key</code>	ABS access key
<code>abs.secret_key</code>	ABS secret key
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	Set to true for ABS-AAD communication to use SSL.
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>
<code>abs.query.interval</code>	NA.
<code>aad.log.level</code>	The log level of AAD log files. The default value is <code>INFO</code> . Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code>
<code>gateway.management.url</code>	NA
<code>gateway.management.username</code>	NA
<code>gateway.management.password</code>	NA
<code>pingaccess.management.url</code>	NA
<code>pingaccess.management.username</code>	NA
<code>pingaccess.management.password</code>	NA

Following is a sample `aad.properties` file:

```
# Automated API Discovery (AAD)

# Automated API Discovery (AAD)

# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod

# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
# or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7I
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7I

### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsJTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSmv5mZllw3WpAsj1oPF3d5Et170Y=
# ABS secret key
abs.secret_key=OBF:AES:RsJTC/tx/sp
+7XXtr8+1rnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
```

```
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74LahxfEgp:dLfcNugFUCcfsglnBXQzflTvwAWiPit8ulseHxi+Z0tk=

### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+lpEqcKQnFG/UN3Ezf0DMa/
kmI=:Az82rlUFftMGpMxF7unelJZUucX191102QgKvHD36vU=
```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `gateway.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

### Generate `aad_master.key`

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```
opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
```

```
config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start AAD after passwords are obfuscated.

Start AAD

**Prerequisite:**

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

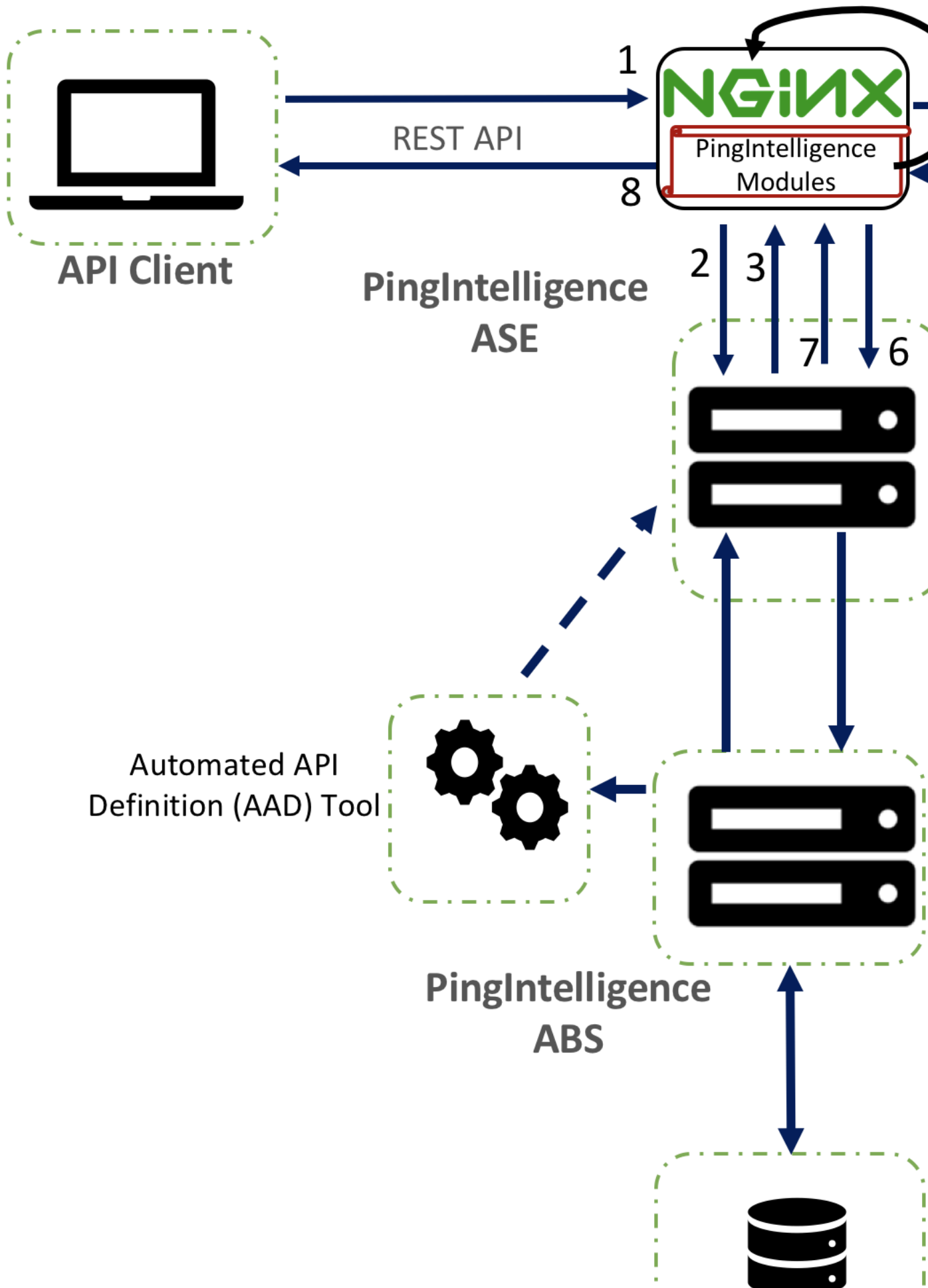
```
bin/stop.sh
Ping Identity Inc.
AAD is stopped.
```

## 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 [Automated PingIntelligence for APIs setup](#) on page 391 or [PingIntelligence manual deployment](#) on page 417

- **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 585

Prerequisites for RHEL 7.6

Complete the following prerequisites before deploying PingIntelligence policy on NGINX:

- **NGINX version:** The PingIntelligence policy modules are compiled 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 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 16

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 extracts the ASE certificate and appends it to the `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 16.04 LTS

Complete the following prerequisites before deploying PingIntelligence policy on NGINX:

- **NGINX version:** The PingIntelligence policy modules are compiled 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 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 16

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 extracts the ASE certificate and appends it to the `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 16.04 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/nginx_ase_integration_module.so
modules/nginx_http_ase_integration_response_module.so
modules/nginx_http_ase_integration_request_module.so
```

The three PingIntelligence modules are:

- a. `nginx_ase_integration_module.so`
- b. `nginx_http_ase_integration_request_module.so`
- c. `nginx_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 582. 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:

```
#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>;
```

**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/nginx_ase_integration_module.so;
load_module modules/nginx_http_ase_integration_request_module.so;
load_module modules/nginx_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 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/nginx_ase_integration_module.so
modules/nginx_http_ase_integration_response_module.so
modules/nginx_http_ase_integration_request_module.so
```

The three PingIntelligence modules are:

- a. `nginx_ase_integration_module.so`
  - b. `nginx_http_ase_integration_request_module.so`
  - c. `nginx_http_ase_integration_response_module.so`
4. Copy the three PingIntelligence modules for Ubuntu to the `modules` directory of NGINX.

```
# cp nginx_ase_integration_module.so /usr/local/nginx/modules
# cp nginx_http_ase_integration_request_module.so /usr/local/nginx/modules
# cp nginx_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/nginx_ase_integration_module.so;
load_module modules/nginx_http_ase_integration_request_module.so;
load_module modules/nginx_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 582. 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:

```

#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>;

```

**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 `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/nginx_ase_integration_module.so;
load_module modules/nginx_http_ase_integration_request_module.so;
load_module modules/nginx_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 15
- [API naming guidelines](#) on page 50
- Adding APIs to ASE in [Defining an API – API JSON configuration file](#) on page 50. You can add individual APIs or you can configure a global API. For more information on global API, see [API discovery](#) on page 203.
- [ABS AI-based security](#) on page 65

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 195
- [API reports using Postman](#) on page 279 .

- [Access the PingIntelligence Dashboard](#) on page 373 .

## API discovery

Automatically build definitions for NGINX Plus APIs in your environment by using PingIntelligence ABS AI Engine. For more information on enabling automatic discovery, see [Enable and disable discovery](#). APIs are discovered when a global API JSON is defined in the ASE. For more information, see [API discovery](#). After the APIs are discovered by the ABS AI Engine, AAD adds the API JSON file to ASE. Install AAD to add discovered API definitions to ASE.

### Install AAD

Download the AAD tool from the [download](#) site. OpenJDK 11 must already be installed on the AAD machine.

Copy the downloaded file to `/opt` directory and run the following command to install:

```
# tar -zxvf aad-4.0.tar.gz
```

The above step installs AAD and creates the following directories:

- `bin` - Contains `start.sh`, `stop.sh` and `status.sh` scripts
- `config` - Contains `aad.properties` file. This file is used to configure AAD
- `data` - For internal use
- `logs` - Contains AAD's logs
- `util` - Contains `thecheck_ports.sh`. Run on the machine with the AAD tool to check ASE and ABS default ports.

The following table describes the AAD configuration parameters:

Property	Description
<code>aad.env</code>	NA
<code>aad.env.dev.fullsync</code>	NA
<code>aad.mode</code>	Set the value to <code>discovery</code> for AAD to work in discovery mode.
<code>abs.host</code>	ABS host IP address
<code>abs.port</code>	ABS host port number
<code>abs.access_key</code>	ABS access key
<code>abs.secret_key</code>	ABS secret key
<code>ase.host</code>	Hostname or IPv4 IP address of the ASE host machine.
<code>ase.port</code>	Port number of the ASE service.
<code>abs.ssl</code>	Set to true for ABS-AAD communication to use SSL.
<code>ase.access_key</code>	The username of ASE. Default value is <code>admin</code>
<code>ase.secret_key</code>	The password of ASE. Default value is <code>admin</code>
<code>abs.query.interval</code>	NA.
<code>aad.log.level</code>	The log level of AAD log files. The default value is <code>INFO</code> . Other possible values are: <code>ALL&lt;DEBUG&lt;INFO&lt;WARN&lt;ERROR&lt;FATAL&lt;OFF</code>
<code>gateway.management.url</code>	NA

gateway.management.username	NA
gateway.management.password	NA
pingaccess.management.url	NA
pingaccess.management.username	NA
pingaccess.management.password	NA

Following is a sample aad.properties file:

```
# Automated API Discovery (AAD)
# Automated API Discovery (AAD)

# AAD deployment environment. Valid values are dev or prod
# In a dev environment AAD adds or updates any API name and API
# metadata changes.
# If aad.env.dev.fullsync=true, then in dev environment AAD deletes any
# API from ASE which is not present at the source.
# In a prod environment, AAD does not delete or update any API name
# change or any API metadata changes.
aad.env=prod

# aad.env.dev.fullsync controls AAD to exactly reflect source APIs in
# ASE or update the source API in ASE. When set to true in a dev
# environment, AAD deletes all APIs from ASE which are not
# present at the source (except url=/ and hostname=* API). Valid values true
# or false
aad.env.dev.fullsync=false

# AAD mode. Valid values discovery, span_port, gateway, and pingaccess
# discovery will pull discovered APIs from ABS
# span_port will pull discovered APIs from ABS
# gateway will pull APIs from Axway API Gateway
# pingaccess will pull APIs from PingAccess
aad.mode=discovery

# AAD query polling interval (minutes) to ABS or Gateway
aad.query.interval=10
# Log level
aad.log.level=INFO
### ASE config
# ASE Host ( hostname or IPv4 address)
ase.host=127.0.0.1
# ASE management port
ase.port=8010
# ASE REST API access key
ase.access_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7E

### ABS config. Only valid if aad.mode=discovery or aad.mode=span_port
# ABS Host ( hostname or IPv4 address )
abs.host=127.0.0.1
# ABS management port
abs.port=8080
# ABS SSL enabled ( true or false )
abs.ssl=true
# ABS access key
abs.access_key=OBF:AES:RsjTC+lxddGqv3XUUV/
YX8iA8kg6Ng==:0vOu0XUVpbvV4AaSvm5mZ1lw3WpAsj1oPF3d5Et170Y=
```

```
# ABS secret key
abs.secret_key=OBF:AES:RsjTC/tx/sp
+7XXtr8+lrnaty1BFug==:78i6bQcdVSavuKm2TXQMOKga/OOEa/ON4RoiUMYu3Rc=

### Axway API Gateway config. Only valid if aad.mode=gateway
# API Manager URL
gateway.management.url=https://127.0.0.1:8075/
# API Manager admin username
gateway.management.username=apiadmin
# API Manager admin password
gateway.management.password=OBF:AES:RMLBOu9/DIVOEAOjYV/
Otw74LahxfEgp:dLfcNugFUCcfsq1nBXQzflTvAWiPit8ulseHxi+Z0tk=

### PingAccess config. Only valid if aad.mode=pingaccess
# Admin URL
pingaccess.management.url=https://127.0.0.1:9000/
# Admin username
pingaccess.management.username=Administrator
# Admin password
pingaccess.management.password=OBF:AES:FevDN+1pEqcKQnFG/UN3Efz0DMA/
kmI=:Az82rlUffftMGpMxF7unelJZUucX191102QgKvHD36vU=
```

### Obfuscate keys and passwords

Using AAD's command line interface, you can obfuscate the keys and passwords configured in `aad.properties`. Following keys and passwords are obfuscated:

- `ase.access_key`
- `ase.secret_key`
- `gateway.management.password`

AAD ships with a default `aad_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `aad_master.key`.

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

### Generate `aad_master.key`

You can generate the `aad_master.key` by running the `generate_obfkey` using the AAD CLI.

```
opt/pingidentity/aad/bin/cli.sh -u admin generate_obfkey -p
Password>
Please take a backup of config/aad_master.key before proceeding.
Warning: Once you create a new obfuscation master key, you should obfuscate
all config keys also using cli.sh obfuscate_keys
Warning: Obfuscation master key file /opt/pingidentity/aad/config/
aad_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/aad/config/
aad_master.key
```

The new `aad_master.key` is used to obfuscate the passwords in `aad.properties` file.

**Note:** After the keys and passwords are obfuscated, the `aad_master.key` must be moved to a secure location from AAD. If you want to restart AAD, the `aad_master.key` must be present in the `/opt/pingidentity/aad/config/` directory.

### Obfuscate keys and passwords

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

```
/opt/pingidentity/aad/bin/cli.sh obfuscate_keys -u admin -p
Password>
Please take a backup of config/aad.properties before proceeding
Enter clear text keys and password before obfuscation.
Following keys will be obfuscated
  config/aad.properties: ase.access_key, ase.secret_key, abs.access_key,
  abs.secret_key and gateway.management.password
Do you want to proceed [y/n]: y
obfuscating /opt/pingidentity/aad/config/aad.properties
Success: secret keys in /opt/pingidentity/aad/config/aad.properties
obfuscated
```

Start AAD after passwords are obfuscated.

Start AAD

#### Prerequisite:

For AAD to start, the `aad_master.key` must be present in the `/opt/pingidentity/aad/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.

To start AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/start.sh
AAD 4.0 starting...
please see /opt/pingidentity/aad/logs/aad.log for more details
```

Stop AAD

To stop AAD, navigate to `/opt/pingidentity/aad/bin` directory and run the following command:

```
bin/stop.sh
Ping Identity Inc.
AAD is stopped.
```

## 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 PoC

---

## Docker - inline PoC deployment

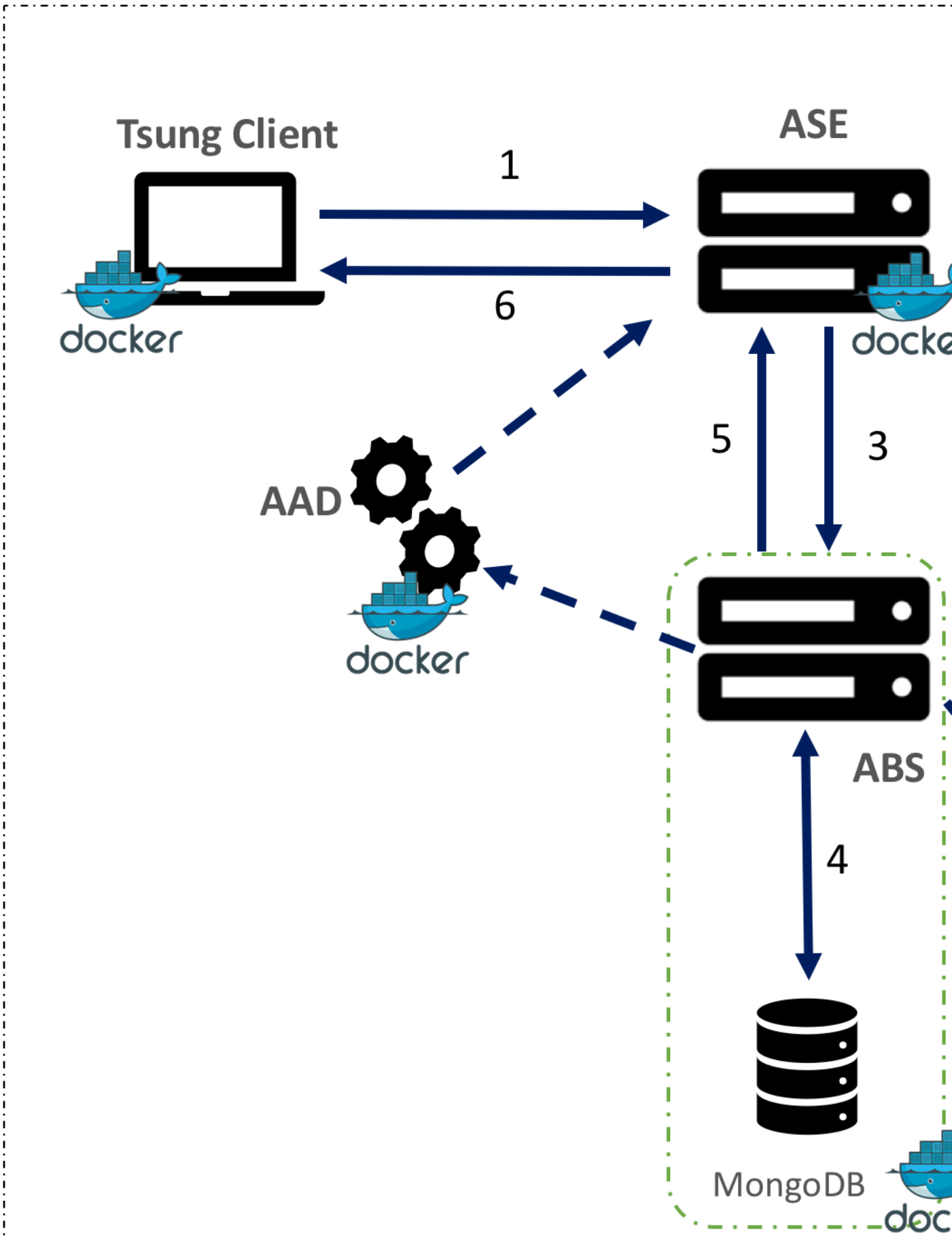
---

### Docker Inline PoC setup

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





**Note:** The Docker images provided are only for evaluation purpose of PingIntelligence for APIs product and should not be used in production deployments.

## Installation requirements

Here is a summary of the software and documentation to download from the [download site](#) as noted below.

### 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. Tsung Traffic Generator
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 Admin Guide](#) - 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:

<b>OS</b>	Ubuntu 16.04
<b>Hardware</b>	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

Download the Docker package from the [download site](#) and save it in the `/opt` directory.

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

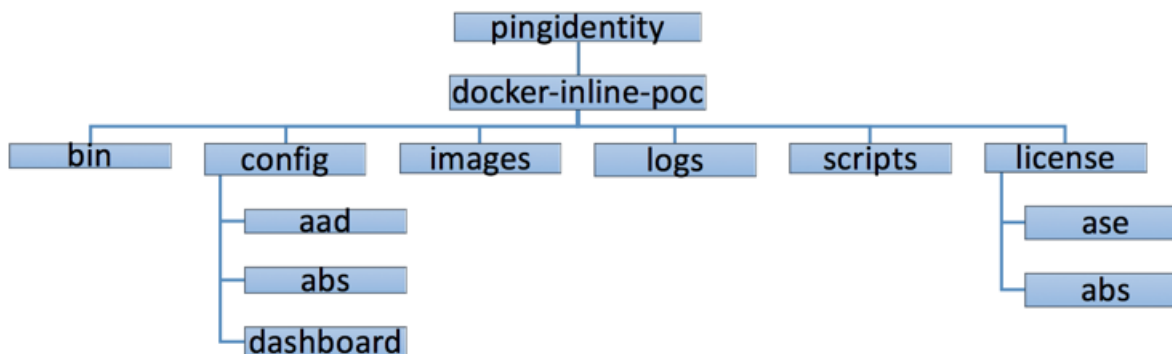
1. Untar the package by running the following command:

```
$sudo tar -xf /opt/docker-inline-poc.tar.gz
```

2. Change the directory to `/opt/pingidentity/docker-inline-poc`.

Directory structure

After you untar the Docker package, the following directory structure is created:



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-inline-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=4.0
IssueDate=2019-05-30
EnforcementType=0
ExpirationDate=2019-11-30
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-inline-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-inline-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=4.0
IssueDate=2019-05-30
```

```
EnforcementType=0
ExpirationDate=2019-11-30
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-inline-poc/license/abs` directory, run the following command:

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

## 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-inline-poc$ sudo ./bin/start.sh install
```

```
root@ip-172-31-25-146:/opt/pingidentity/docker-inline-poc# ./bin/start.sh
install
Mon Oct 14 05:30:57 UTC 2019 : loading ASE image
Loaded image: pingidentity/ase:4.0.2
Mon Oct 14 05:31:00 UTC 2019 : loading ABS image
Loaded image: pingidentity/abs:4.0.2
Mon Oct 14 05:31:07 UTC 2019 : loading AAD image
Loaded image: pingidentity/aad:4.0.2
Mon Oct 14 05:31:11 UTC 2019 : loading Dashboard image
Loaded image: pingidentity/dashboard:4.0.2
Mon Oct 14 05:31:25 UTC 2019 : loading client image
Loaded image: pingidentity/client:4.0.2
Mon Oct 14 05:31:29 UTC 2019 : loading server image
Loaded image: pingidentity/server:4.0.2
Mon Oct 14 05:31:32 UTC 2019 : loading mongo image
Loaded image: pingidentity/mongo:4.2.0
Mon Oct 14 05:31:37 UTC 2019 : Installation completed successfully
```

## Setup the PoC environment

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

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

```
Mon Oct 14 05:33:03 UTC 2019 : Starting setup scripts
Creating network pingidentity_net
Creating config pingidentity_abs_license
Creating config pingidentity_shop_api_json
Creating config pingidentity_shop_books_api_json
Creating config pingidentity_shop_electronics_api_json
Creating config pingidentity_decoy_api_json
Creating config pingidentity_ase_license
Creating service pingidentity_client
Creating service pingidentity_mongo
Creating service pingidentity_abs
Creating service pingidentity_ase
Creating service pingidentity_dashboard
Creating service pingidentity_aad
Creating service pingidentity_server
Mon Oct 14 05:33:26 UTC 2019 : 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-inline-poc#sudo ./bin/start.sh setup
Tue Oct 15 16:55:12 UTC 2019 : Starting setup scripts
Creating network pingidentity_net
open /opt/pingidentity/docker-inline-poc/license/ase/PingIntelligence.lic: no such file or directory
Tue Oct 15 16:55:13 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-inline-poc$sudo ./bin/start.sh training
```

```
root@vortex-108:/opt/pingidentity/docker-inline-poc$sudo ./bin/start.sh
training
Wed Oct 16 13:44:25 IST 2019 : Starting model training scripts
Wed Oct 16 13:44:25 IST 2019 : 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-inline-poc$sudo ./bin/start.sh attack
```

```
root@vortex-108:/opt/pingidentity/docker-inline-poc$sudo ./bin/start.sh
attack
Wed Oct 16 15:40:51 IST 2019 : Starting attack scripts
Wed Oct 16 16:40:51 IST 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.

Execute the following script to test API deception:

```
root@vortex-108:/opt/pingidentity/docker-inline-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

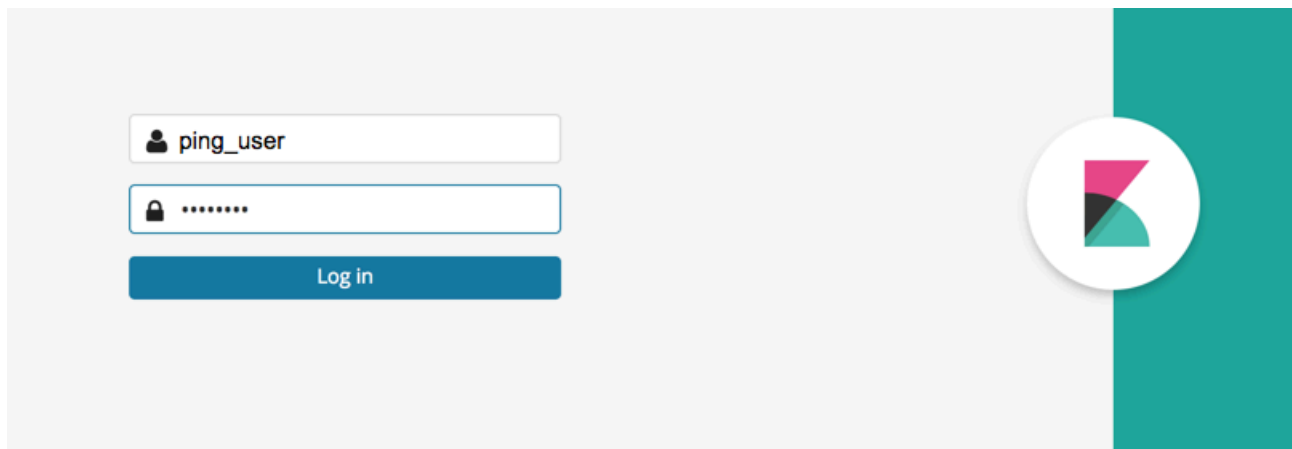
```

## 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](#) on page 203.

## View dashboard reports

Access the main dashboard with a browser at this URL: <https://<host-machine>ip:5601/app/kibana#/dashboard/pingintelligence> . In the above URL, <ip:port> is the IP address of the host machine where the PingIntelligence PoC is set up. The default username and password of for logging is: ping\_user/change



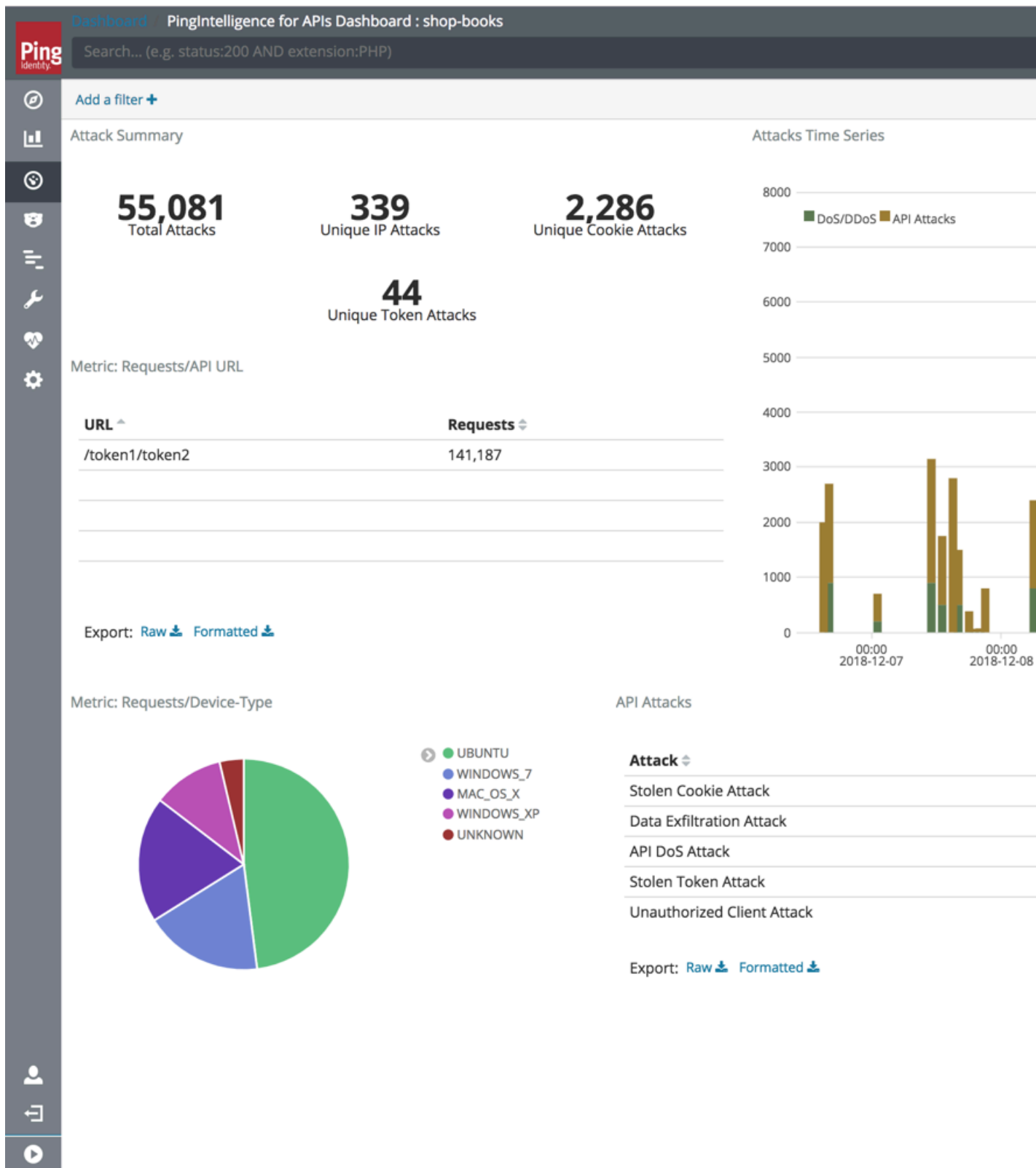
To navigate to the main dashboard, click **PingIntelligence for APIs Dashboard**. To navigate to a specific API's dashboard, click **PingIntelligence for APIs Dashboard: <api name>** for example, **PingIntelligence for APIs Dashboard: shop-books**.

The screenshot displays the Ping Identity Dashboard interface. On the left is a dark sidebar with the Ping Identity logo at the top. Below the logo are several menu items, each with an icon: Discover (compass), Visualize (bar chart), Dashboard (globe, currently selected and highlighted in dark grey), Timelion (clock), Dev Tools (wrench), Management (gear), Ping Dashboard User (person), Logout (door with arrow), and Collapse (left arrow). The main content area is light grey and features a search bar at the top with the text "Search...". Below the search bar is a list of dashboard items, each with a checkbox and a label: "Name" with a small upward arrow, "Decoy API Dashboard", "PingIntelligence for APIs Dashboard", "PingIntelligence for APIs Dashboard : shop" (with a yellow arrow pointing to it from the right), "PingIntelligence for APIs Dashboard : shop-book", and "PingIntelligence for APIs Dashboard : shop-elect".

The PingIntelligence for APIs Dashboard provides information on attack activity across all APIs, and separate Dashboards are also available for each individual API and Decoy APIs.

The Dashboard for each API looks like the following:



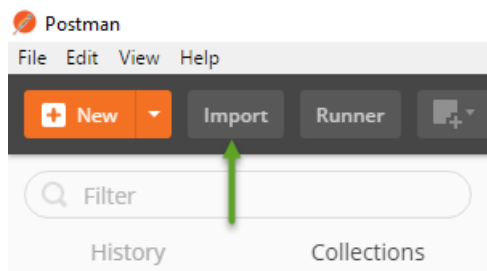



### ABS detailed reporting

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_4.0_Environment` and `ABS_4.0_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\_4.0\_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**.

The screenshot shows the Postman application interface. On the left, the 'Collections' tab is active, displaying a list of reports under the 'ABS\_3.2\_Reports' collection. The 'Administration' report is selected. The main area shows a REST client configuration for a GET request to a URL containing a variable `{{System_Admin}}`. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a JSON object with the following structure:

```

1 - {
2   "company": "ping identity",
3   "name": "api_admin",
4   "description": "This report contains status information on all",
5   "across_api_prediction_mode": true,
6 -  "api_discovery": {
7     "subpath_length": "1",
8     "status": true
9   },
10 -  "apis": [
11 -    {
12      "api_name": "apikeyquery",
13      "host_name": "*",
14      "url": "/apikeyquery",
15      "api_type": "decoy-incontext",
16      "creation_date": "Fri Dec 07 16:45:05 IST 2018",
17      "servers": 4,
18      "protocol": "https",
19      "cookie": "",
20      "token": false,
21      "training_started_at": "Fri Dec 07 16:47:00 IST 2018",
22      "training_duration": "1 hour",
23      "prediction_mode": true
24    },
25 -    {
26      "api_name": "ipapp",
27      "host_name": "*",
28      "url": "/ipapp",
29      "api_type": "decoy-incontext",
30      "creation_date": "Fri Dec 07 16:45:05 IST 2018",
31      "servers": 4,
32      "protocol": "https",
33      "cookie": "",
34      "token": false,
35      "training_started_at": "Fri Dec 07 16:47:00 IST 2018",
36      "training duration": "1 hour".

```

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
Wed Oct 23 18:45:42 IST 2019 : Starting stop scripts
Removing service pingidentity_abs
Removing service pingidentity_ase
Removing service pingidentity_client
Removing service pingidentity_dashboard
Removing service pingidentity_server
Removing config pingidentity_shop_electronics_api_json
Removing config pingidentity_shop_books_api_json
Removing config pingidentity_decoy_api_json
Removing config pingidentity_shop_api_json
Removing network pingidentity_net
Wed Oct 23 19 18:45:42 IST 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`

## Cloud PoC service deployment

---

### PingIntelligence Cloud PoC 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:

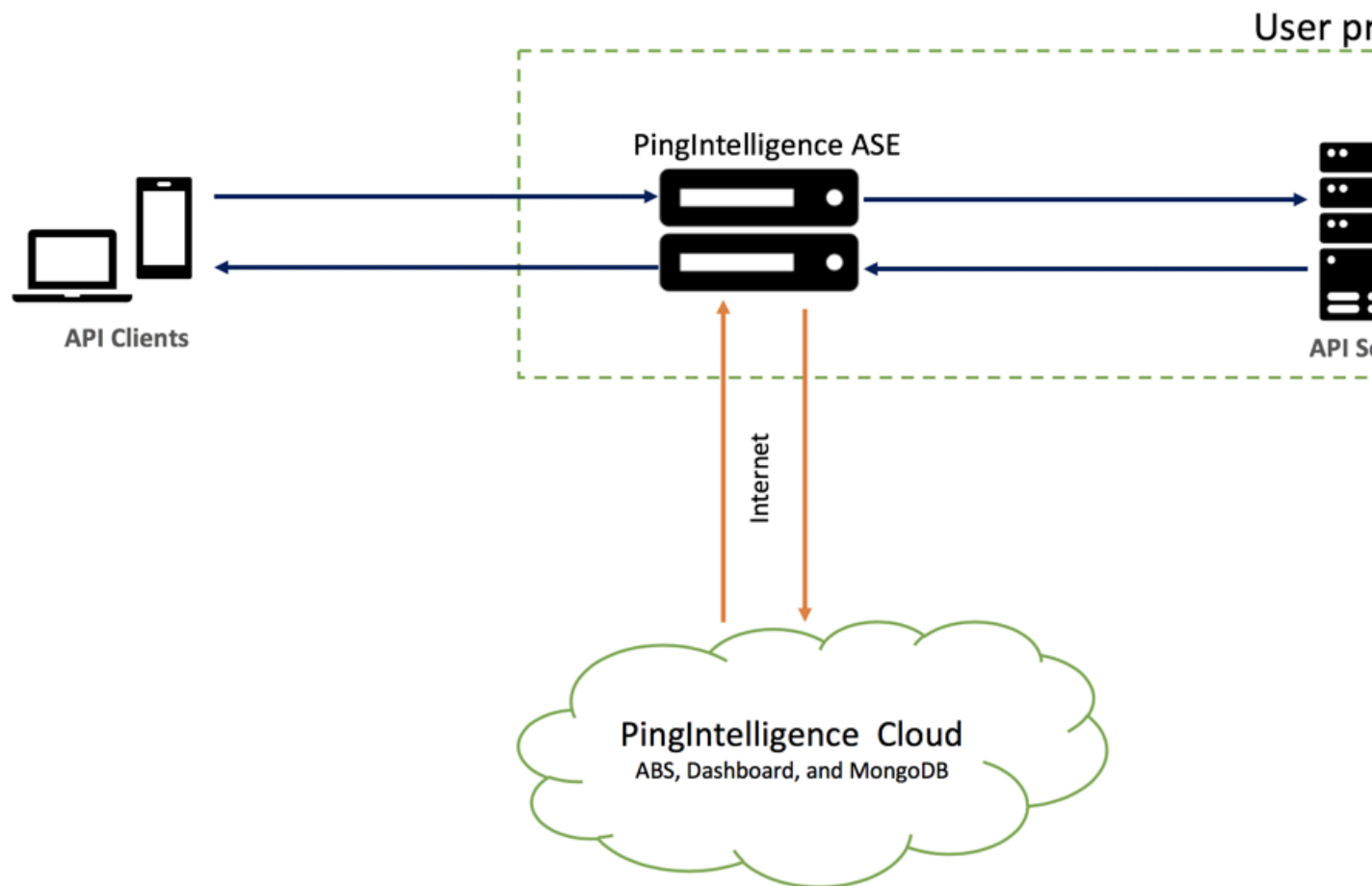
- PingIntelligence ABS, Dashboard, and MongoDB are hosted as a cloud service managed by Ping Identity
- PingIntelligence SaaS ASE is deployed in your API environment.

PingIntelligence Cloud service can be deployed in two modes:

- Inline mode
- Sideband mode

#### Inline mode

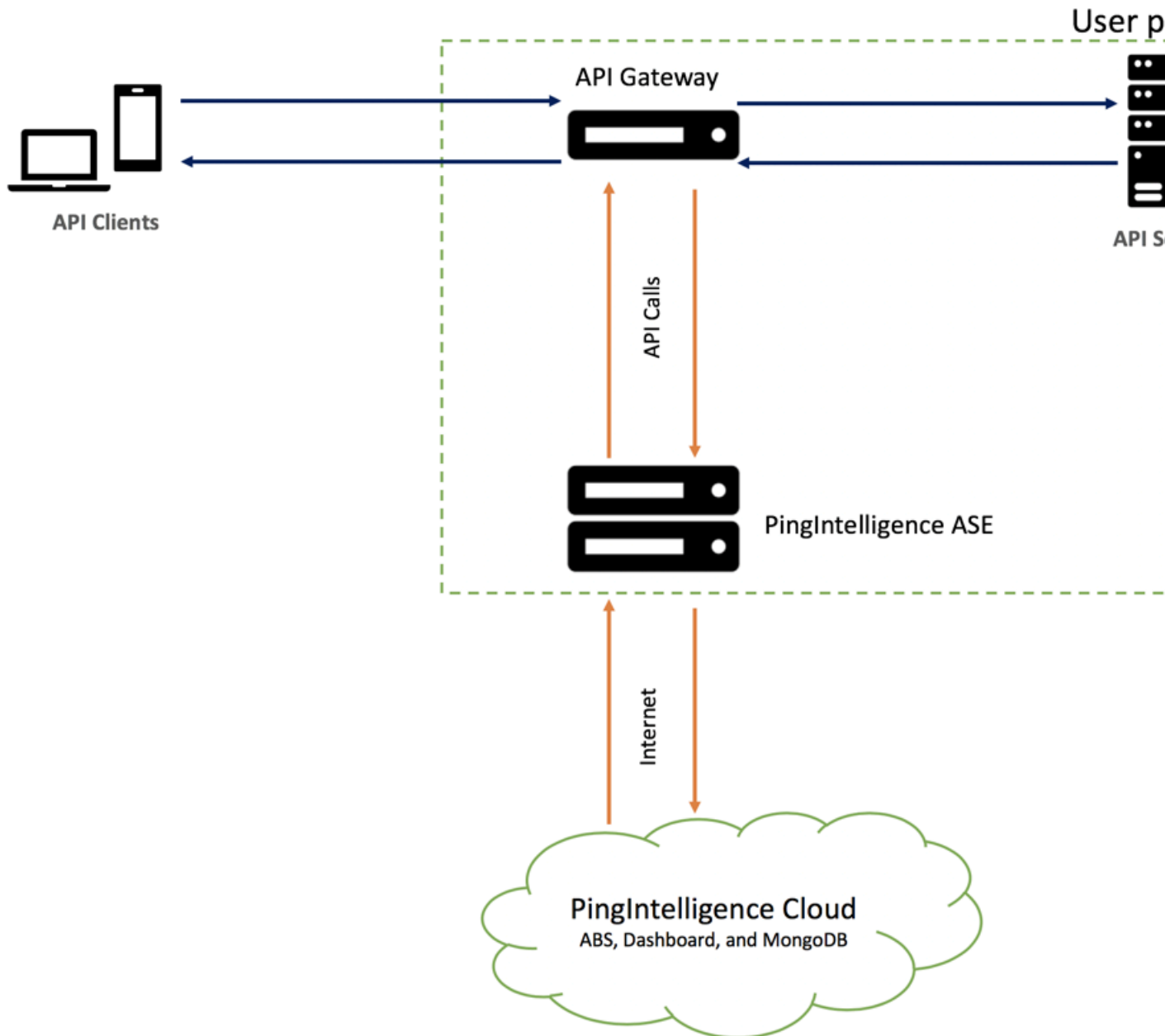
In inline mode, ASE receives API client traffic and routes the traffic to API servers. It can be deployed behind an existing load balancer, such as AWS ELB. In inline mode, ASE terminates SSL connections from API clients and then routes the API requests to the target APIs – running on an API Gateway 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. Ping currently supports sideband policies on the following platforms: Apigee API Gateway, Axway API Gateway, and PingAccess.

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 information 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.

### Install ASE using Docker image

You can also install ASE using the Docker image provided to you.

**Server requirements:** The set up requires one machine which hosts the ASE image. The server requirement for the machine is as follows:

- Ubuntu 16
- 4 CPUs, 16 GB RAM, 500 GB Storage

**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 package and Configure `setup.conf`:

Download the Docker package from the link provided to you and save it in the `/opt` directory. Complete the following steps before Installing and loading the Docker images:

- Untar the package by running the following command:

```
root@ip-172-31-28-18:/opt # tar -xf docker-poc.tar.gz
```

- Change the directory to `/opt/pingidentity/docker-poc`. Configure the `config/setup.conf` file:

The setup has two ASE modes – inline or sideband. Depending on the mode set in the `setup.conf` file, ASE will be start in inline or sideband mode.

Following is a sample of the `setup.conf` file:

```
# Provide setup details here
# Mode: inline or sideband
mode="sideband"

# ABS IP/Hostname
abs_ip="<ABS IP address from email you received from PingIntelligence>"

#ABS port
abs_port="<Port number from email you received from PingIntelligence>"
abs_access_keys="<Access key from email you received from PingIntelligence>"
abs_secret_keys="<Secret key from email you received from PingIntelligence>"
```

### Install and load Docker images:

To install and load Docker images, run the following command on your host Ubuntu 16 machine after you have configured `setup.conf`.

This command loads and installs the Docker images from the images directory.

```
root@ip-172-31-17-178:/opt/pingidentity/docker-poc# ./bin/start.sh install
ABS IP: a7ff77849d45a11e99bb322-2128927072.us-east-1.elb.amazonaws.com:8080
Wed Sep 11 09:59:52 UTC 2019 : loading ASE image
Loaded image: pingidentity/ase:4.0.1
Wed Sep 11 09:59:58 UTC 2019 : Installation completed successfully
```

### Install ASE license for ASE:

PingIntelligence ASE will require a valid license to start. The license file is named `PingIntelligence.lic`. Complete the following steps:

Copy the provided ASE license file to the following path:

```
/opt/pingidentity/docker-inline-poc/license/ase/PingIntelligence.lic
```

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=4.0
IssueDate=2019-09-01
EnforcementType=0
ExpirationDate=2019-12-30
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-inline-poc/license/ase` directory, run the following command:

```
# grep 'Module' /opt/pingidentity/docker-inline-poc/license/ase /
PingIntelligence.lic
Module=ASE
```



**Start the ASE Docker container and setup:** To start the Docker Containers and setup, run the following command on the host Ubuntu 16 machine:

```
root@ip-172-31-17-178:/opt/pingidentity/docker-poc/bin# ./start.sh setup
ABS IP: a7ff77849d45a11e99bb322-2128927072.us-east-1.elb.amazonaws.com:8080
Wed Sep 11 13:22:57 UTC 2019 : Starting setup scripts
Creating network pingidentity_net
Creating config pingidentity_ase_license
Creating service pingidentity_ase
Wed Sep 11 13:22:58 UTC 2019 : Setup successful Wed Sep 11 13:22:58 UTC
2019 : Setup successful
```

**Restarting ASE:** If you need to restart ASE from within the docker container (for any configuration change), make sure that ASE is restarted within 300-seconds of stopping, else the container restarts with a new container ID.

**Stop ASE:** Enter the following command to stop ASE:

```
root@ip-172-31-17-178:/opt/pingidentity/docker-poc/bin# ./bin/stop.sh
Wed Sep 11 16:23:49 UTC 2019 : Starting stop scripts
Removing service pingidentity_ase
Removing config pingidentity_ase_license
Removing network pingidentity_net
Wed Sep 11 16:23:49 UTC 2019 : Stop successful
```

## 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.

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

## 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

## 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
+ploQ:jN6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0

; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw:zzi77GIFF5xkQJccjlrIVWU
+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=
```

## Start and stop ASE

### Start 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.

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

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 4.0...
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:

- [Apigee Integration](#)
- [Axway Integration](#)
- [PingAccess Integration](#)

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

## 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: <code>ws</code> - WebSocket ; <code>http</code> - HTTP
url	The value of the URL for the managed API. You can configure up to 3 levels of API. <code>"/shopping"</code> - name of a 1 level API <code>"/shopping/electronics/phones"</code> - 3 level API <code>"/"</code> - entire server (used for ABS API Discovery or load balancing)
hostname	Hostname for the API. The value cannot be empty. <code>"*"</code> matches any hostname.
cookie	Name of cookie used by the backend servers.
oauth2_access_token	When <code>true</code> , ASE captures OAuth2 Access Tokens. When <code>false</code> , ASE does not look for OAuth2 Tokens. Default value is <code>false</code> . For more information, see <a href="#">Configuring OAuth2 Token</a> .
apikey_qs	When API Key is sent in the query string, ASE uses the specified parameter. For more information, see <a href="#">Configuring API Keys</a> .
apikey_header	When API Key is part of the header field, ASE uses the specified parameter. For more information, see <a href="#">Configuring API Keys</a> .
login_url	Public URL used by a client to connect to the application.
health_check	When <code>true</code> , enable health checking of backend servers. When <code>false</code> , no health checks are performed. Ping Identity recommends setting this parameter as <code>true</code> .
health_check_interval	The interval in seconds at which ASE sends a health check to determine the status of the backend server.
health_retry_count	The number of times ASE queries the backend server status after not receiving a response.
health_url	The URL used by ASE to check backend server status.
server_ssl	When set to <code>true</code> , ASE connects to the backend API server over SSL.

**Servers:**

host

The IP address or hostname and port number of each backend server

port

See [REST API Protection from DoS and DDoS](#) for information on options

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 from

client\_spike\_threshold

See [WebSocket API Protection from DoS and DDoS](#) for information on options

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 precedence

methods\_allowed

List of accepted REST API methods. Possible values are:

GET, POST, PUT, DELETE, HEAD

content\_type\_allowed

List of content types allowed. Multiple values cannot be listed. For example,

**Decoy Config**

When decoy\_enabled is set to true, decoy sub-paths function as decoy

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 returns

response\_def response\_message

response\_message is the response message (for example OK) that ASE returns

decoy\_subpaths

decoy\_subpaths is the list of decoy API sub-paths (for example shop, admin)

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
```

**Download and install AAD**

PingIntelligence provides AAD tool to automatically add APIs to ASE. The AAD tool works in the following two modes. You can refer to this [section](#) to configure AAD.

- In the *discovery* mode, AAD fetches the APIs discovered by ABS and deploys to ASE. By default, discovery is enabled in ABS. For ABS to discover APIs, a global API should be configured in PingIntelligence ASE.
- In the *gateway* mode (for *PingAccess* and *Axway*), AAD connects to the gateway and fetches the API definition and deploys to ASE. To use AAD with an API gateway, you should the management host IP address and port number. This information is configured in AAD's properties file.

Follow the instructions at [this link](#) to download and install AAD.

 **Note:** For detailed information on ASE administration, see the [ASE admin guide](#)

## 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.

The dashboard URL that you received in the email from Ping is used to load the PingIntelligence dashboard. You are supplied with the following username and password **Dashboard user and password**. Use this username and password

to view the main PingIntelligence dashboard as shown in the screenshot below:

The screenshot displays the PingIntelligence for APIs Dashboard. The top navigation bar includes the Ping Identity logo, a search bar with the text "> Search... (e.g. status:200 AND extension:PHP)", and the page title "Dashboard PingIntelligence for APIs Dashboard". A sidebar on the left contains a "Dashboard" menu item. The main content area features two summary cards: "82 IP Blacklist" and "18 Cookie Blacklist". Below these is a section titled "Attacks" containing a table with columns "API" and "URL".

API	URL
shop-books	/shopapi-books
shop-electronics	/shopapi-electronics
shop	/shopapi
decoy	/decoy

For more information on accessing and using Dashboard, see [Access the 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 Docker toolkit

---

## PingIntelligence Docker toolkit

---

PingIntelligence for APIs provides a Docker toolkit using which you can create Docker images of the various PingIntelligence products, tools, and MongoDB. The output of the Docker toolkit are Docker images. The docker toolkit should be run either on a RHEL 7.6 or a Ubuntu 16 LTS machine.

**Prerequisites:** [Download](#) the following PingIntelligence products and tools. You also need to download few third-party products.

- **Download products:**
  - PingIntelligence ASE 4.0.2
  - PingIntelligence ABS 4.0.2
  - PingIntelligence AAD 4.0.2
  - PingIntelligence Dashboard 4.0.2
  - PingIntelligence AMT
  - MongoDB 4.2
  - OpenJDK 11.0.2
  - 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 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 docker-toolkit-4.0.2.tar.gz
```

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

Directory	Description
bin	Contains the <code>build.sh</code> script to build the Docker images
config	Contains the <code>docker.conf</code> file to configure the base image name and the base image operating system
data	For internal use

external	Contains the third-party software: <ul style="list-style-type: none"> <li>▪ MongoDB 4.2</li> <li>▪ Elasticsearch 6.8.1</li> <li>▪ Kibana 6.8.1</li> <li>▪ OpenJDK 11.0.2</li> </ul>
images	Contains the created Docker images using the <code>build.sh</code> script
lib	For internal use
license	Contains the <code>ase</code> and <code>abs</code> directory to copy the respective license files. <p><b>Note:</b> You can build the images without adding the license files to the <code>ase</code> and <code>abs</code> directory. If you build the Docker images without the license file in <code>ase</code> and <code>abs</code> directory, then you need to map or mount the license file in the following exact location:</p> <ul style="list-style-type: none"> <li>▪ ASE: <code>/opt/pingidentity/ase/config/PingIntelligence.lic</code></li> <li>▪ ABS: <code>/opt/pingidentity/abs/config/PingIntelligence.lic</code></li> </ul>
logs	Contains the log files
software	Contains PingIntelligence ASE, ABS, AAD, AMT, 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
- AAD

- AMT
- MongoDB

It is a good practice to obfuscate the various keys and password in ASE, ABS, AAD, AMT, 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](#)
- AAD - [Obfuscate keys and passwords](#)
- AMT [Obfuscate key and password](#)
- 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	<code>ase.tar.gz</code>
ABS	<code>abs.tgz</code>
Dashboard	<code>dashboard.tar.gz</code>
AAD	<code>aad.tar.gz</code>
AMT	<code>amt.tar.gz</code>

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	<code>elasticsearch.tar.gz</code>
OpenJDK 11.0.2	<code>openjdk11.tar.gz</code>
Kibana	<code>kibana.tar.gz</code>
MongoDB	<code>mongodb.tgz</code>

**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:4.0.2
Image saved to /home/ubuntu/docker-setup/images/pingidentity_ase.tar
Creating build context for abs
Creating Image
Image created with tag pingidentity/abs:4.0.2
Image saved to /home/ubuntu/docker-setup/images/pingidentity_abs.tar
Creating build context for aad
Creating Image
Image created with tag pingidentity/aad_amt:4.0.2
Image saved to /home/ubuntu/docker-setup/images/pingidentity_aad_amt.tar
```

```

Creating build context for dashboard
Creating Image
Image created with tag pingidentity/dashboard:4.0.2
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`, `aad_amt`, `dashboard`, `mongo`.

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

```

# docker image ls
REPOSITORY                                TAG          IMAGE ID          SIZE
pingidentity/dashboard                    4.0.2       b172c856756a     32
  minutes ago                             1.24GB
pingidentity/ase                          4.0.2       a29b708d9467     2
  hours ago                               329MB
pingidentity/mongo                        4.2.0       0460e8717341     47
  hours ago                               485MB
pingidentity/aad_amt                      4.0.2       04ed26b185cd     47
  hours ago                               540MB
pingidentity/abs                          4.0.2       0ddda6aa3755     47
  hours ago                               771MB

```

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 Sep 18 18:39 ..
-rw----- 1 root root  782182400 Sep 21 10:18 pingidentity_abs.tar
-rw----- 1 root root  600428544 Sep 21 10:18 pingidentity_aad_amt.tar
-rw----- 1 root root  495038976 Sep 21 10:20 pingidentity_mongo.tar
-rw----- 1 root root  339437568 Sep 23 06:57 pingidentity_ase.tar
-rw----- 1 root root 1302484480 Sep 23 08:08 pingidentity_dashboard.tar
drwxr-xr-x  2 root root      4096 Sep 23 08:08 .

```

The Docker images that are built do not have 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, AAD, and MongoDB.

**ASE Environment Variables:** The following table lists the ASE environment variables and the values:

Environment	Value	Usage
ASE_MODE	inline/ sideband	ASE can be deployed either in inline mode or sideband mode. For more information, see the <a href="#">ASE admin guide</a> .
ASE_ENABLE_CLUSTER	true/false	Set the value to <code>true</code> to enable ASE cluster.
ASE_ENABLE_ABS	true/false	Set the value to <code>true</code> to enable ABS.
ASE_PEER_NODE	<IP or hostname>:port	ASE cluster peer node's IP address and port number

ASE_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

**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

**MongoDB Environment Variables:** The following table lists the MongoDB environment variables and the values:

Environment	Value	Usage
ABS_ACCESS_KEY	string	The access key for the ABS admin user. For more information, see <a href="#">ABS users</a>
ABS_SECRET_KEY	string	The secret key for the ABS admin user. For more information, see <a href="#">ABS users</a>
ABS_ACCESS_KEY_RESTRICTED	string	The access key for the restricted user. For more information on restricted user, see <a href="#">ABS users</a> .
ABS_SECRET_KEY_RESTRICTED	string	The secret key for the restricted user. For more information on restricted user, see <a href="#">ABS users</a> .
MONGO_USERNAME	string	MongoDB username
MONGO_PASSWORD	string	MongoDB password
ATTACK_INITIAL_TRAINING	INTERVAL	The attack training period
ATTACK_UPDATE_INTERVAL	INTERVAL	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 <a href="#">API discovery</a> for more information.
API_DISCOVERY_INITIAL_PERIOD	INTERVAL	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	INTERVAL	The time period in hours in which ABS reports the newly discovered APIs
API_DISCOVERY_SUBPATHS	NUMBER	The number of subpaths that are discovered in an API. The maximum value is 3.
WIRED_TIGER_CACHE_SIZE_GB	FILE_SIZE_GB	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
KIBANA_PASSWORD	string	Password for Kibana
ELASTIC_PASSWORD	string	Password for Elasticsearch
PINGUSER_PASSWORD	string	Password for pinguser
PINGADMIN_PASSWORD	string	Password for ping_admin
ABS_ACCESS_KEY	string	The access key for the ABS admin user. For more information, see <a href="#">ABS users</a>
ABS_SECRET_KEY	string	The secret key for the ABS admin user. For more information, see <a href="#">ABS users</a>
ABS_HOST	string	IP address of ABS host
ENABLE_XPACK	string	Configures whether x-pack is installed. Default value is <code>true</code> .
ENABLE_UI	string	Configures whether Dashboard is displayed or not. The default value is <code>true</code> .
ENABLE_SYSLOG	string	Configures whether Dashboard sends syslog messages to the syslog server. The default value is <code>false</code> .  <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p><b>ⓘ Important:</b> <code>ENABLE_SYSLOG</code> and <code>ENABLE_UI</code> both cannot be <code>false</code> at the same time.</p> </div> <p>When <code>ENABLE_SYSLOG</code> environment variable is passed to the container, <code>SYSLOG_HOST</code> and <code>SYSLOG_PORT</code> should also be passed. These are to configure the syslog server and port number.</p>
ABS_RESTRICTED	<code>true/false</code>	Set to <code>true</code> if you want to use ABS restricted user. For more information on restricted user, see <a href="#">ABS users</a> .

**AAD Environment Variables:** The following table lists the AAD environment variables and the values:

Environment	Value	Usage
AAD_MODE	string	The AAD mode
AAD_ENVIRONMENT	string	Configures the AAD environment. The default is production ( <code>prod</code> ) environment.
AAD_ENV_DEV_FULL_SYNC	string	Configures the synchronization of APIs. The default value is <code>false</code> .
ABS_HOST	string	IP address of ABS host
ABS_ACCESS_KEY	string	The access key for the ABS admin user. For more information, see <a href="#">ABS users</a>
ABS_SECRET_KEY	string	The secret key for the ABS admin user. For more information, see <a href="#">ABS users</a>
ASE_HOST	string	IP address of ASE host
ASE_ACCESS_KEY	string	Access key to connect to ASE
ASE_SECRET_KEY	string	Secret key to connect to ASE

GATEWAY_MANAGEMENT_URL		Gateway management URL. Configure if you have Axway API gateway in your deployment.
GATEWAY_MANAGEMENT_USERNAME		Gateway management username. Configure if you have Axway API gateway in your deployment.
GATEWAY_MANAGEMENT_PASSWORD		Gateway management password. Configure if you have Axway API gateway in your deployment.
PINGACCESS_MANAGEMENT_URL		PingAccess management URL. Configure when mode is set to pingaccess.
PINGACCESS_MANAGEMENT_USERNAME		PingAccess management username. Configure when mode is set to pingaccess.
PINGACCESS_MANAGEMENT_PASSWORD		PingAccess management password. Configure when mode is set to pingaccess.

## 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. AAD and AMT
5. 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:4.0.2
```

**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 ASE_MODE="inline"
-e\
ASE_ENABLE_CLUSTER="true" -e ASE_PEER_NODE="ase:8020" -e
ASE_ENABLE_ABS="true" -e\
ASE_ABS_ENDPOINT="abs:8080" -e ABS_ACCESS_KEY="new_abs_ak" -e\
```

```
ABS_SECRET_KEY="new_abs_sk" --shm-size=1g pingidentity/ase:4.0.2'
```

**Launch the second ASE node in ASE cluster:** Run the following command with some sample environment variables to launch the ABS container:

```
# docker run -d --name ase1 --link abs:abs --link ase:ase --hostname ase1 -e
-e\
ASE_MODE="inline" ASE_ENABLE_CLUSTER="true" -e ASE_PEER_NODE="ase:8020" -e -
e\
ASE_ENABLE_ABS="true" ASE_ABS_ENDPOINT="abs:8080" -e
  ABS_ACCESS_KEY="new_abs_ak" -e\
ABS_SECRET_KEY="new_abs_sk" --shm-size=1g pingidentity/ase:4.0.2
```

**Launch AAD container:** Run the following command with some sample environment variables to launch the ABS container:

```
docker run -d --name aad --link abs:abs --link ase:ase --hostname aad -e
  AAD_MODE="discovery" -e\
ABS_HOST="abs" -e ABS_ACCESS_KEY="new_abs_ak" -e ABS_SECRET_KEY="new_abs_sk"
-e ASE_HOST="ase" -e\
ASE_ACCESS_KEY="admin" -e ASE_SECRET_KEY="admin" -e AAD_ENVIRONMENT="dev" -e
\
AAD_ENV_DEV_FULLSYNC=true pingidentity/aad_amt:4.0.2
```

**Launch Dashboard:** Run the following command with some sample environment variables to launch the ABS container:

```
# docker run -d --name dashboard --link abs:abs --hostname dashboard -e\
KIBANA_PASSWORD="new_kibana_password" -e
  ELASTIC_PASSWORD="new_elastic_password" -e\
PINGUSER_PASSWORD="new_ping_user_password" -e
  PINGADMIN_PASSWORD="new_ping_admin_password" -e\
ABS_RESTRICTED_USE_ACCESS="true" -e ABS_ACCESS_KEY="new_abs_ak_ru" -e\
ABS_SECRET_KEY="new_abs_sk_ru" -e ABS_HOST="abs" pingidentity/
dashboard:4.0.2
```

## 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	Usage
ASE	8080	HTTP data plane
	8443	HTTPS data plane
	8010	Management port
	8020	Cluster port
ABS	8080	API server port
	9090	Access log upload port
Dashboard	5601	Kibana port
MongoDB	27017	MongoDB port



# PingIntelligence App Notes

---

## Dashboard deployment using NGINX

---

NGINX is a Web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. PingIntelligence for APIs Dashboard supports using NGINX proxy functionality to authenticate users accessing Kibana. Create users in NGINX by using `htpasswd` and then enable access to PingIntelligence Dashboard.

### Install NGINX

To install NGINX, use the [NGINX installation](#) URL. The following steps install NGINX on a RHEL machine:

#### 1. Add NGINX repository:


```
# cat /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/rhel/7/$basearch/
gpgcheck=0
enabled=1
```

#### 2. Install NGINX:

```
# yum install -y nginx
```

#### 3. Install httpd-tools:

```
# yum install -y httpd-tools
```

 **Note:** `httpd-tools` is installed to create users to connect to Kibana.

### Setting up a user

Add a NGINX user with the following command:

```
# htpasswd -c /etc/nginx/.htpasswd ping_user
New password:
Re-type new password:
Adding password for user ping_user
```

In the above example, username `ping_user` is added with a password which can be used to log in to Kibana.

NGINX can also be configured to use LDAP authentication. For more information, see [Using NGINX Plus and NGINX to Authenticate Application Users with LDAP](#)

PingIntelligence Dashboard supports using two Elasticsearch deployment options:

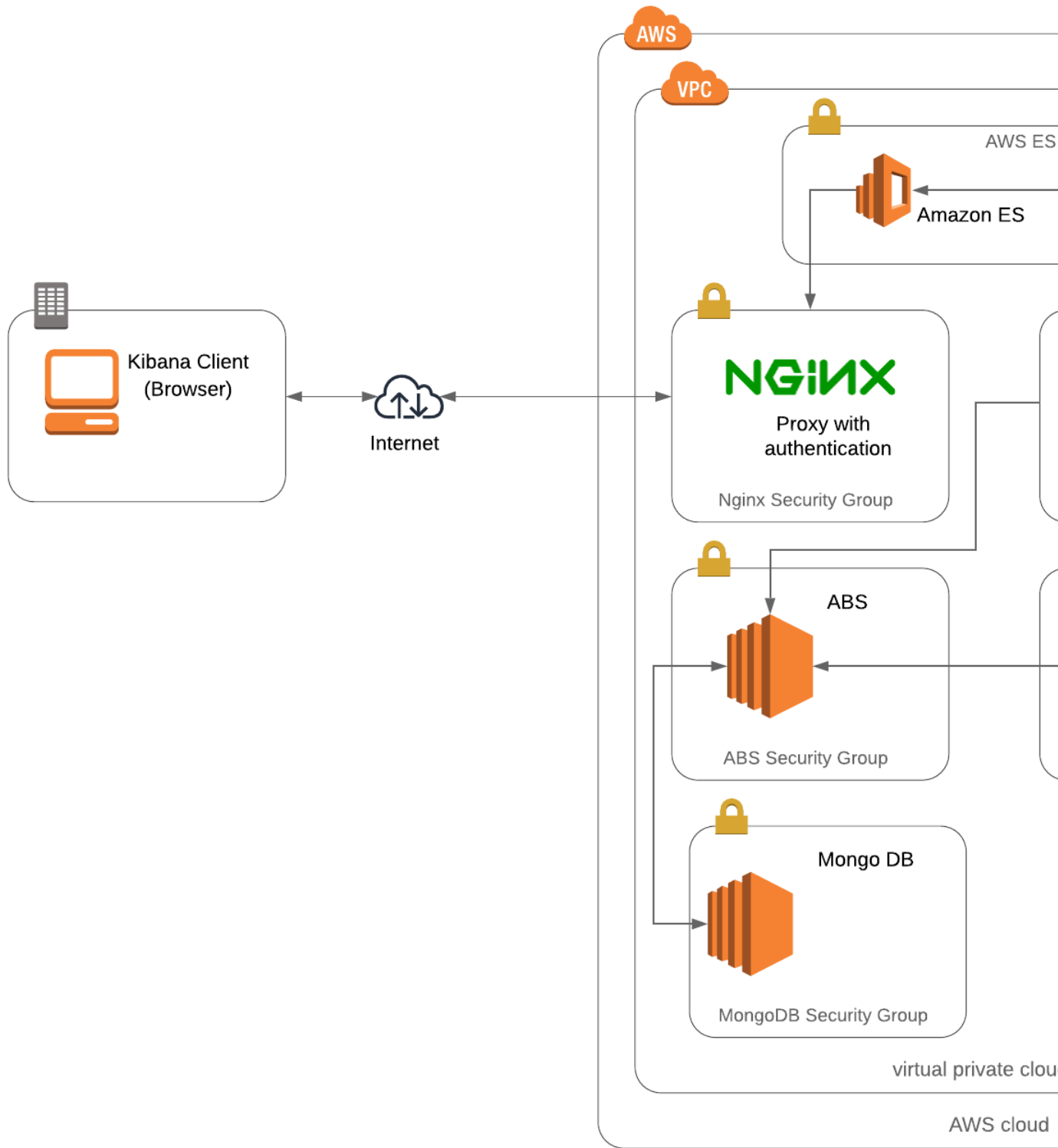
- [Using Amazon Elasticsearch](#)
- [On-premise Elasticsearch](#)

## PingIntelligence Dashboard using Amazon ES

This app note describes how to use Amazon ES to provide encrypted access to a PingIntelligence Dashboard. It is assumed that PingIntelligence is installed and configured. For more information on installing PingIntelligence, see [PingIntelligence automated or manual deployment guides](#).

This app note describes a solution similar to an AWS solution documented in [How to Control Access to Your Amazon Elasticsearch Service Domain](#)

The following diagram provides a high-level overview of PingIntelligence for APIs using Amazon ES for its Elasticsearch service and using NGINX proxy service to authenticate access to the Kibana dashboard.



**Note:** The Amazon ES is not open for public access. It is only available in a specific VPC where both PingIntelligence Dashboard node and Nginx node can access it.

## Prerequisites to setup

Setting up Amazon ES for PingIntelligence Dashboard requires the following prerequisites to be complete:

- PingIntelligence Dashboard should be installed and configured. For more information on Dashboard, see the [Dashboard admin guide](#)
- Amazon ES is configured. For more information, see [Amazon ES documentation](#)
- Read and understand the NGINX Proxy configuration. For more information, see [Nginx configuration examples](#)

## Amazon ES and PingIntelligence Dashboard configuration

Complete the Amazon ES configuration using the information available at [Sizing Amazon ES Domains](#).

After completing Amazon ES the configuration for PingIntelligence, gather the following information:

- VPC endpoint
- Kibana URL
- Access policy authorizing the Dashboard node to write to the Elasticsearch instance
- Kibana version number, for example, 6.4.2 or later

The following screenshot shows the detail input for Amazon ES configuration. Make a note of the VPC endpoint and Kibana

# pingintelligence

**Configure cluster**

**Modify access policy**

**Manage t...**

**Overview**

**Cluster health**

**Instance health**

**VPC**

**Domain status** Active

**Elasticsearch version** 6.4

**VPC endpoint** [https://vpc-pingintelligence-](https://vpc-pingintelligence-...)

**Domain ARN** arn:aws:es:ap-... :12770...

**Kibana** [https://vpc-pingintelligence-](https://vpc-pingintelligence-...)

URL.

**Note:** Make sure that no Kibana browsers are open.

### Check connectivity between Dashboard and Amazon ES

To check the connectivity between the Dashboard and Amazon ES, use the `curl` command as shown in the following example. `ES_Endpoint_URL` is the VPC endpoint URL as shown in the screenshot above.

```
[root@ip-172-31-24-54 ~]# curl https://ES_Endpoint_URL/_cluster/health
{
  "cluster_name": "377367197819:pitest1",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "active_primary_shards": 0,
  "active_shards": 0,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100.0
}
```

### Configure dashboard.properties file

Edit the `/opt/pingidentity/dashboard/config/dashboard.properties` file to configure the following values:

- `es.url`: valid Elasticsearch URL
- `es.username`: blank if Dashboard node can access Elasticsearch without any username or password
- `es.password`: blank if Dashboard node can access Elasticsearch without any username or password
- `kibana.version`: 6.4.2 or 6.4.3

Following is a sample `dashboard.properties` file

```
# Dashboard properties file
# ABS Hostname/IPv4 address
abs.host=172.31.24.54
# 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:NuCmDdihqRNlRtU8SMKMoLaSpJviT4kArw==HHuA9sAPDiOen3VU
+qp6kMrkgNjAwnKO6aa8pMuZkQw=
# ABS secret key
abs.secret_key=OBF:AES:NuBmDcAhCeNlPBDmyMX+685CBe8R3/STVA==:BIfH
+FKmL5cNaIdrfVuy5hIYjimqh7Rnf3bv9hW0+4=
# ABS query polling interval (minutes)
abs.query.interval=10
# ABS query offset (minutes. minimum value 30 minutes)
abs.query.offset=30
# elasticsearch URL
es.url=https://ES_Endpoint_URL
# elasticsearch username. User should have manage_security privilege
es.username=
```

```
# elasticsearch user password
es.password=
# kibana version
kibana.version=6.4.2
# Log level
dashboard.log.level=INFO
```

## NGINX configuration

This section describes configuring the NGINX proxy server to authenticate Dashboard access.

Set up NGINX to proxy login requests

Update `nginx.conf` settings to configure a proxy service for authentication. In the following example, we are configuring a HTTPS based connection:

```
user nginx;
worker_processes 1;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;

    keepalive_timeout 65;

    server {
        listen *:443 ssl;
        ssl_certificate /api/certs/server.crt;
        ssl_certificate_key /api/certs/server.key;
        ssl_dhparam /api/certs/dhparam.pem;
        server_name NGINX_HOSTNAME;
        location / {
            auth_basic "ping_intelligence";
            auth_basic_user_file /etc/nginx/.htpasswd;
            proxy_set_header Authorization "";
            proxy_pass http://KIBANA_ENDPOINT_URL/;
        }
    }
}
```

Note that `KIBANA_ENDPOINT_URL` is configured without `_plugin/kibana/`. Restart NGINX for the changes to take effect.

## Access PingIntelligence Dashboard

You can access the Kibana endpoint by using the following URL:

[https://nginx\\_ip/](https://nginx_ip/)

Replace the `nginx_ip` with public IP of you NGINX server. Use the username and password configured using `htpasswd`.

## Dashboard using on-premise Elasticsearch

Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents and Description.

Kibana is an open source data visualization plug-in for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

Complete the following steps to use on-premise Elasticsearch:

Download packages

Download the following two packages:

- Elasticsearch
- Kibana

### Elasticsearch

Complete the following step to download Elasticsearch:

```
#wget https://artifacts.elastic.co/downloads/elasticsearch/
elasticsearch-6.6.0.rpm
```

### Kibana

Complete the following step to download Kibana:

```
#wget https://artifacts.elastic.co/downloads/kibana/kibana-6.6.0-x86_64.rpm
```

Install Elasticsearch and Kibana

### Install Elasticsearch

Install Elasticsearch on a Java machine by completing the following step.

```
# rpm -ivh elasticsearch-6.6.0.rpm
```

### Install Kibana

Install Kibana by completing the following step:

```
# rpm -ivh kibana-6.6.0-x86_64.rpm
```

Configure Elasticsearch and Kibana

### Configure Elasticsearch

Edit the `elasticsearch.yml` file to specify the data and logs path

```
# vim /etc/elasticsearch/elasticsearch.yml

path.data: /var/lib/elasticsearch
```

```
path.logs: /var/log/elasticsearch
```

## Configure Kibana

Edit the `kibana.yml` file to specify the server and elasticsearch URL.

```
#vim /etc/kibana/kibana.yml

server.host: "172.16.40.244"
elasticsearch.url: "http://localhost:9200"

# ./bin/kibana-plugin install file:///opt/pingidentity/dashboard/plugins/
pingstyling-4.0.zip
```

## Start Elasticsearch and Kibana

Run the following command to start Elasticsearch and Kibana:

```
# systemctl start elasticsearch.service
# systemctl start kibana.service
```

Verify that Elasticsearch and Kibana has started:

```
# systemctl status elasticsearch.service
# systemctl status kibana.service
```

## Configure nginx.conf

Configure the `nginx.conf` file to add the Kibana URL:

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid       /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include        /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
    ,
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile    on;

    keepalive_timeout  65;

    server {
```



```
listen *:443 ssl;
ssl_certificate /api/certs/server.crt;
ssl_certificate_key /api/certs/server.key;
ssl_dhparam /api/certs/dhparam.pem;
server_name NGINX_HOSTNAME;
location / {
    auth_basic "ping_intelligence";
    auth_basic_user_file /etc/nginx/.htpasswd;
    proxy_set_header Authorization "";
    proxy_pass http://KIBANA_IP:5601/;
}
}
```

Access PingIntelligence dashboard

Access the Kibana endpoint by using the following URL:

[https://nginx\\_ip/](https://nginx_ip/)

Replace the `nginx_ip` with public IP of you NGINX server. Use the username and password configured using `htpasswd`.