

PingIntelligence

ABS 3.2.2 Release Notes

MongoDB critical reads: ABS 3.2.2 release provides an enhancement for critical read operations in MongoDB. Critical read operations are now done through MongoDB's primary node.

ASE 3.2.2 Release Notes

Keep alive connection flag for ASE sideband

ASE 3.2.2 introduces `enable_sideband_keepalive` variable with a default value as `false` in `ase.conf` file. This variable is only applicable for [sideband deployments of ASE](#) (`mode=sideband` in `ase.conf`). When set to `true`, ASE sends a keep-alive in response header for the TCP connection between API gateway and ASE. With the default `false` value, ASE sends a connection close in response header for connection between API gateway and ASE. Following is a snippet of `ase.conf` file with the new variable:

```
; enable connection keepalive for requests from gateway to ase.
; This setting is applicable only in sideband mode.
; Once enabled ase will add 'Connection: keep-alive' header in response
; Once disabled ase will add 'Connection: close' header in response
enable_sideband_keepalive=false
```

3.2.1 Release Notes

- ASE
- ABS
- AAD
- Dashboard

ASE

- **Install ASE as a non-root user:** ASE can be installed as a non-root user with a customer defined ASE installation path. For more information, see [Install ASE Software](#)
- **Purge script:** ASE purge script can now be run as a non-root user. Purge script now deletes the `controller.log` files. For more information, see [Purge log files](#)

ABS

- **Install ABS as a non-root user:** ABS AI Engine can be installed as a non-root user with a customer defined. For more information, see [Install ABS AI engine](#)
- **Disk usage limit:** A new parameter `percentage_diskusage_limit` is added to the `abs.init` file. This parameter governs the disk usage limit at which ABS stops accepting access log files from ASE. The default value is set to 80%. This parameter is also reported in the ABS Admin API. For more information, see [Admin REST API](#)
- **Email alert** in the log file: All the email alerts are now logged in the `abs.log` file. For more information, see [ABS alerts](#)
- **Update.sh script:** The ABS `update.sh` script can now update additional `abs.init` file parameters:
 - `window_length`
 - `url_limit`
 - `discovery_subpath`
 - `percentage_diskusage_limit`

For more information, see [Update the training variables](#)

- **Enhanced Admin API:** ABS Admin API now displays the ABS scale-up and scale-down settings:

```
"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"
  }
}
```

For more information, see *Admin REST API*

AAD

Purge script: AAD now provides a purge script which is located in the `/pingidentity/aad/util` directory. AAD purge script removes or archives `aad.log` files. For more information, see [Purge AAD log files](#)

Dashboard

Purge script: Dashboard provides the following two new purge scripts:

- **purge_elasticsearch:** The Elasticsearch purge script deletes or archives Elasticsearch files. For more information, see *Purge data from Elasticsearch*
- **Purge:** The purge script removes or archives `dashboard.log` files. For more information, see *Purge dashboard logs*

Features at a glance

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

- Enhanced AI-based attack detection including content scraping, cross-API attack detection, and extended time frame attacks, for example, slow login, extended data exfiltration.
- API Discovery support for OAuth tokens
- Additional MongoDB database high availability deployment options with replica set support

ABS features

- OAuth2 token-based metrics and forensics reporting – reports which show token activity on an API and detailed reporting of all activity for a specific token.
- 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.
- Advanced Artificial Intelligence and machine learning software detects and reports on suspected cyber attacks including complex attacks requiring correlation across multiple clients.

- Examples of detected attacks include data exfiltration, stolen cookie attacks, multiple types of API Distributed Denial of Service (DDoS) attacks, API memory attacks, and brute force login attacks.
- Attack detection based on OAuth token, Cookie, or IP address activity.
- Automatic sending of attack lists 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.
- Detection and reporting of anomalous behavior to be evaluated as a potential attack.
- Discovering and reporting on valid and invalid URLs used to access each secured API.
- Comprehensive reporting on API traffic including:
 - API activity - client information, API and URL usage, metrics, anomalies, backend errors, attack types, and decoy traffic.
 - ASE detected attack reporting – Detailed reports on hackers blocked by ASE for invalid API activity and access of Decoy APIs. Provides client identification including OAuth V2 token, cookie, or IP address information.
 - API Key based activity reporting – API key activity report provides details on all API activity from each API key.
 - REST API interface for access to API JSON reports which are also accessible by customer management systems for API monitoring.
- PingIntelligence for APIs Dashboard (installed separately) uses Kibana graphs to provide summary information on metrics, anomalies, and attacks for each API.
- Stateless ABS clustering with redundant MongoDB databases for scale and high availability.
- Flexible deployment options on standard Linux servers (RHEL 7, Ubuntu 16 LTS) running JVMs across various environments such as Containers, VMs, bare metal servers, and Clouds.
- Automatic generation of alerts to notify operations of errors or capacity issues.
- Password obfuscation to keep passwords encrypted
- 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.

ABS Introduction

API Behavioral Security Engine (ABS) 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 provides the following functionality:

- Collection and consolidation of access logs from API Security Enforcer nodes
- Machine learning algorithms to identify anomalies and attacks
- Detection of attacks from HTTP(s) and WebSocket(s) traffic
- Optional sending of attack lists to API Security Enforcer which blocks client access
- Centralized database for storing machine learning output

- 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:** 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 APIsDashboard:** 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.

ABS 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](#)
- [Start and Stop ABS](#)
- [Change default settings](#)
- [ABS users for API reports](#)
- [ABS directory structure](#)
- [Obfuscate passwords](#)
- [Configure SSL](#)
- [Import existing CA-signed certificates](#)
- [ABS ports](#)
- [ABS cluster](#)
- [ABS initial configuration](#)
- [Connect ABS to API Security Enforcer](#)
- [ABS logs](#)
- [Purge the processed access logs from ABS](#)
- [Purge MongoDB data](#)
- [Configure email notifications](#)
- [ABS REST API format](#)
- [Admin REST API](#)

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: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.

Parent topic: [ABS administration](#)

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:

```
$ /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 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
```

Parent topic: [ABS administration](#)

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>" } )
```

Parent topic: [ABS administration](#)

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, and API keys 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, and IP 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"
});
```

Parent topic: [ABS administration](#)

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.
<code>logs</code>	Stores all ABS related logs.
<code>lib</code>	For internal use. Do not change anything in this directory.
<code>bin</code>	Contains various scripts to start and stop ABS.

	 Note: 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"> • Check and Open MongoDB Default Port • Purge the Processed Access Logs from ABS • Purge ABS Data from MongoDB • <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.

Parent topic: [ABS administration](#)

Obfuscate passwords

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

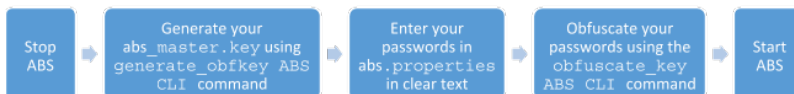
- `mongo_password`
- `jks_password`
- `email_password`

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.

Parent topic: [ABS administration](#)

Configure SSL

ABS supports only TLS 1.2 and requires Java 8 u161 and later. 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.

Parent topic: [ABS administration](#)

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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3F
```

Enter the following command:

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

For example:

```
# keytool -importkeystore -destkeystore abs.jks -srckeystore abs.p12
-srcstoretype PKCS12 -alias exampleCAcertificate
Importing keystore abs.p12 to abs.jks...
```

```
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
```

- Copy the `abs.jks` file created in step 2 to `/opt/pingidentity/abs/config/ssl` directory.
- Start ABS by entering the following command:

```
# /opt/pingidentity/abs/bin/start.sh
Starting API Behavioral Security 3.2...
please see /opt/pingidentity/abs/logs/abs/abs.log for more details
```

Parent topic:[ABS administration](#)

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.

Parent topic:[ABS administration](#)

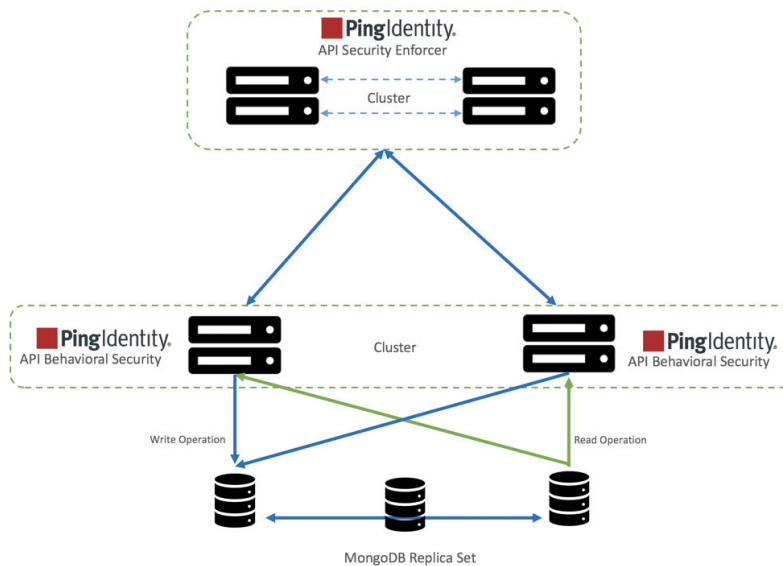
ABS cluster

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.

Parent topic: [ABS administration](#)

ABS initial configuration

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
<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>log_level</code>	Log detail captured. The default is INFO.

	Additional options - DEBUG, ERROR.
mongo_rs	Comma separated MongoDB replica set nodes IP addresses and port numbers. A maximum of three nodes can be configured.
metadata_dbname	The MongoDB metadata database name. The default value is abs_metadata .
data_dbname	The MongoDB data database name. The default value is abs_data .
mldata_dbname	The MongoDB machine learning database name. The default value is abs_mldata
mongo_username	Username of MongoDB.  Note: Required for MongoDB authentication
mongo_password	MongoDB password
runaway_time	Maximum time in minutes to wait for a job to finish. The default value is 120 minutes.
queue_size	Do not change the value of this parameter. The default is 10.
dashboard_node	When true , designated as a dedicated Reporting or Dashboard node. This ABS node does not process log data or participate in an ABS cluster. The default value is false .  Note: Multiple nodes can be Reporting or Dashboard nodes.
system_memory	Memory size in MB allocated to run machine learning jobs. Recommended to be at least 25% of system memory.
system_json_size	Memory size in KB allocated for API JSON files. The default is 500 KB.
enable_emails	Enable (true

) or disable (false) ABS email notifications.
sender_email	Email address used for sending email alerts and reports.
receiver_email	Email address notified about alerts and reports.
smtp_port	Port number of SMTP server.
smtp_host	Hostname of SMTP server.
jks_password	The password of the JKS Keystore. ABS ships with a default obfuscated password. You can reset the password and obfuscate it. This password should be the same that you would use in importing your CA-signed certificate .

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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCull
# 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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3FFCull
# 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
# 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

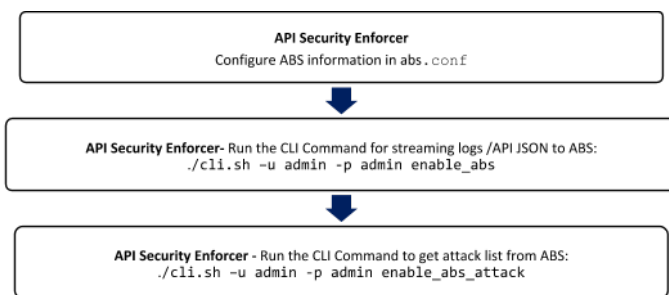
```

Parent topic:[ABS administration](#)

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: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=
```

The `access_key` and `secret_key` are the keys that were defined in the `abs_init.js` file when configuring MongoDB.



Note: To connect an API Security Enforcer cluster to ABS, configure the `abs.conf` file on any API Security Enforcer in the cluster and run the CLI commands. This ensures all the API Security Enforcer nodes in the cluster will be updated to connect with ABS.

If ABS is running in cluster mode, choose the IP address and port from any ABS node to add to the `abs.conf` file in API Security Enforcer.

Dataflow

API Security Enforcer connects to the ABS node defined in `abs.conf` to obtain available ABS IP addresses (step 1). In stand-alone mode, ABS sends the only IP address. In cluster mode, ABS sends the IP addresses of all available ABS nodes to API Security Enforcer.

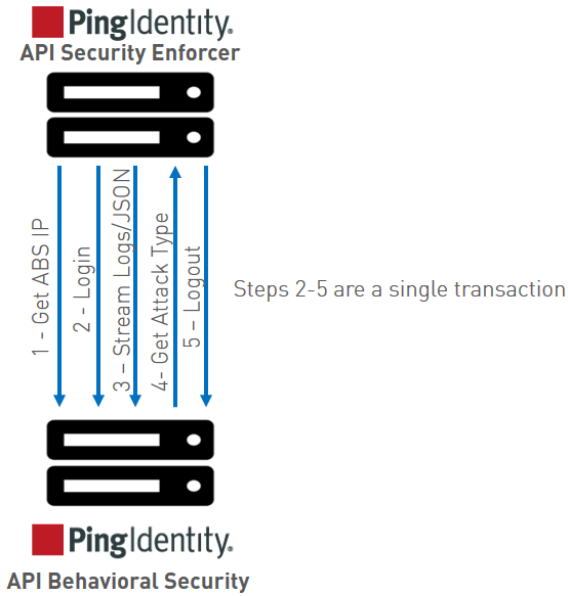
After API Security Enforcer receives the IP address, it establishes a session with ABS by sending the secret and access keys (step 2). After successful authentication, API Security Enforcer streams the access log files and API JSON files to the ABS node (step 3). After sending the files, it receives the attack lists (only available if blocking is activated for API Security Enforcer) from ABS (step 4). When the transaction is complete, API Security Enforcer logs out from ABS (step 5).

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

The following diagrams show the API Security Enforcer – ABS Dataflow.

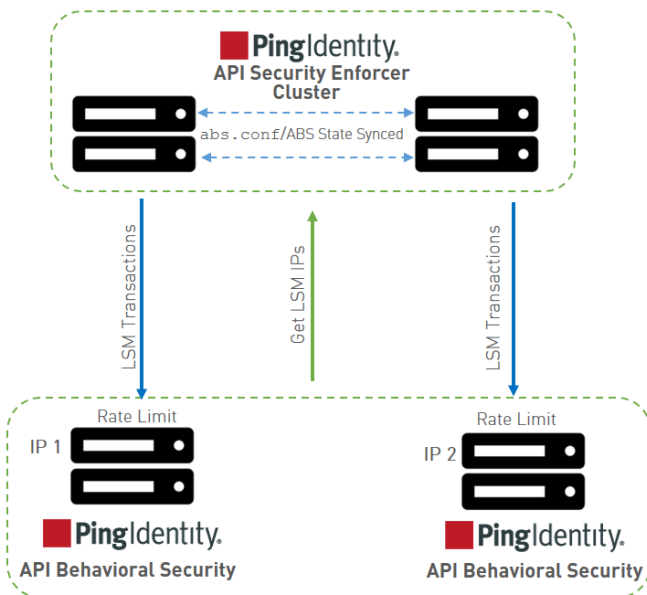
Stand-alone mode

In stand-alone mode, a single MongoDB node is used for both read and write operations. A stand-alone mode of deployment is only recommended for testing purposes.



Cluster mode

In cluster mode, API Security Enforcer nodes synchronize the `abs.conf` file as well as the state of each ABS node. The ABS cluster nodes do not communicate among themselves. Each node records its status in MongoDB and reads about the state of other nodes from the database.



Parent topic: [ABS administration](#)

ABS logs

The active ABS log file `abs.log` is located in the logs directory and rotated every 24-hours at midnight local time. The rotated log files append timestamps to the name and follow the naming convention of `abs.log.<yyyy>-<mm>-<dd>` (for example, `abs.log.2018-11-24`). Here is an example:

```
-rw-r--r--. 1 root root 68K Oct 24 23:59 abs.log.2018-11-24
-rw-r--r--. 1 root root 68K Oct 25 23:59 abs.log.2018-10-25
```

```
-rw-r--r--. 1 root root 68K Oct 26 23:59 abs.log.2018-10-26
-rw-r--r--. 1 root root 158K Oct 27 23:59 abs.log.2018-10-27
-rw-r--r--. 1 root root 32K Oct 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:

```
2018-10-28 11:16:45 INFO - starting abs periodic actions
2018-10-28 11:16:45 INFO - MongoDB heartbeat success
2018-10-28 11:16:45 INFO - notification node not set.
2018-10-28 11:16:45 INFO - training period 1 hours.
2018-10-28 11:16:45 INFO - system threshold update interval 1 hour(s).
2018-10-28 11:16:45 INFO - api discovery interval 1 hour(s).
2018-10-28 11:16:45 INFO - subpath limit: 100
2018-10-28 11:16:45 INFO - ABS started successfully...
2018-10-28 11:17:45 INFO - MongoDB heartbeat success
2018-10-28 11:19:45 ERROR - MongoDB heartbeat failure
```

Parent topic:[ABS administration](#)

Purge the processed access logs from ABS

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



Note: When the `purge` script is run, the processed access log files are permanently deleted from the `/opt/pingidentity/abs/data` directory. Always backup the files before deleting.

Located in the `/opt/pingidentity/abs/util` directory, the `purge` script deletes logs older than the specified number of days. Run the script using the ABS command line. For example:

```
/opt/pingidentity/abs/util/purge.sh -d 3
In the above example, purge.sh deletes all access log files older than 3 days. Here is sample output.
/opt/pingidentity/abs/util/purge.sh -d 3
This will delete the data in /opt/pingidentity/abs/data which is older than 3 days.
Are you sure (yes/no): yes
removing /opt/pingidentity/abs/data/
2018-04-10-11_21/9k2unv5l2bsgurneot3s3pmt03/ : last changed at Mon Jan 10
11:32:31 IST 2018
removing /opt/pingidentity/abs/data/2018-04-10-11_21/
ilq67a3g5sve2pmpk271o37c/ : 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.

Parent topic: [ABS administration](#)

Purge MongoDB data

Purge MongoDB data

The ABS MongoDB purge script dumps and/or deletes processed ABS 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 machine.

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`, `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

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:

```
/opt/pingidentity/mongo/purge_mongo.sh -u absuser -p abs123 --db abs_data --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 --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 --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.

Parent topic:[ABS administration](#)

Configure email notifications

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.

```
#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](#)
- [ABS reports](#)

Parent topic:[ABS administration](#)

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 or Filesystem cross a threshold value.
- **ABS Utilization:** alert sent when ABS cannot accept more logs since resources are fully utilized.
- **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.
- **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.

- **DDoS attack alert:** ABS sends alerts for multi-client Login Attacks and for API Memory Attack Type 2. 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"
  }
]
});
```

Parent topic: [Configure email notifications](#)

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
- Number of successful API requests
- Size of access logs processed
- Number of Attacks and Anomalies reported

Parent topic: [Configure email notifications](#)

ABS REST API format

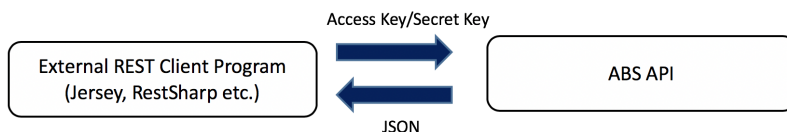
ABS provides external REST APIs which are used to access JSON reports providing deep insight into the following:

- 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
<code>api_name</code>	The API name to query for results.
<code>earlier_date</code>	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.
<code>later_date</code>	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.

Parent topic: [ABS administration](#)

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 API 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": false,
  "api_discovery": {
    "status": false
  },
  "apis": [
    {
      "api_name": "apikeyquery",
      "host_name": "*",
      "url": "/apikeyquery",
      "api_type": "decoy-incontext",
      "creation_date": "Wed Feb 06 19:36:22 IST 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": "",
      "token": false,
      "training_started_at": "training not started yet",
      "training_duration": "24 hours",
      "prediction_mode": false
    },
    {
      "api_name": "apikeyheader",
      "host_name": "*",
      "url": "/apikeyheader",
      "api_type": "decoy-incontext",
      "creation_date": "Wed Feb 06 19:36:22 IST 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": "",
      "token": false,
      "training_started_at": "training not started yet",
      "training_duration": "24 hours",
      "prediction_mode": false
    }
  ],
}
```

```

{
  "api_name": "atmapp",
  "host_name": "*",
  "url": "/atmapp",
  "api_type": "decoy-incontext",
  "creation_date": "Wed Feb 06 19:36:22 IST 2019",
  "servers": 4,
  "protocol": "https",
  "cookie": "",
  "token": false,
  "training_started_at": "training not started yet",
  "training_duration": "24 hours",
  "prediction_mode": false
},
{
  "api_name": "pubatmapp",
  "host_name": "*",
  "url": "/pubatmapp",
  "api_type": "decoy-incontext",
  "creation_date": "Wed Feb 06 19:36:22 IST 2019",
  "servers": 4,
  "protocol": "https",
  "cookie": "JSESSIONID",
  "token": false,
  "training_started_at": "training not started yet",
  "training_duration": "24 hours",
  "prediction_mode": false
}
],
"abs_cluster": {
  "abs_nodes": [
    {
      "node_ip": "192.168.11.165",
      "os": "Red Hat Enterprise Linux Server release 7.4 (Maipo)",
      "cpu": "24",
      "memory": "62G",
      "filesystem": "76%",
      "bootup_date": "Tue Feb 05 16:12:41 IST 2019"
    }
  ],
  "mongodb_nodes": [
    {
      "node_ip": "192.168.11.168",
      "status": "up"
    }
  ]
},
"ase_logs": [
  {
    "ase_node": "13eea2fc-64d0-4c51-b663-b1093b0bf7a5",
    "last_connected": "Wed Feb 06 19:41:07 IST 2019",
    "logs": {
      "start_time": "Wed Feb 06 19:36:26 IST 2019",
      "end_time": "Wed Feb 06 19:41:07 IST 2019",

```

```

        "gzip_size": "27.51MB"
    }
}
],
"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"
    }
}
}
}

```

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.

Parent topic:[ABS administration](#)

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 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](#).

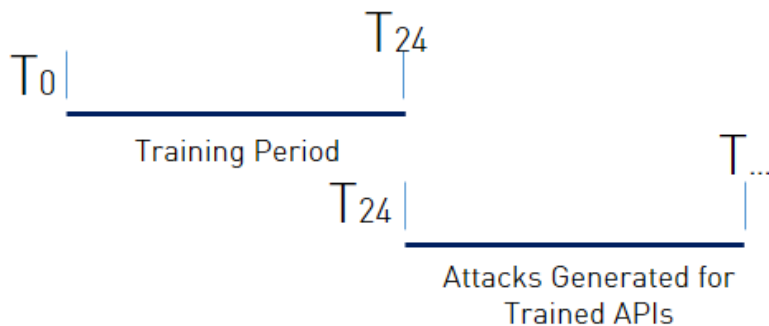
ABS must be trained on all APIs before they can be secured. Whenever a new API is added, ABS automatically trains itself before looking for attacks.

- [Training the ABS model](#)
- [AI Engine training variables](#)
- [Training period status](#)
- [Update the training variables](#)
- [Tune thresholds for false positives](#)
- [Disable attack detection](#)

Training the ABS model

ABS can be trained in a live or a staging 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 starts detecting attacks. In this case, no database migration is required. 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.

Parent topic: [AI Engine training](#)

AI Engine training variables

PingIntelligence AI training depends on a set of parameters configured in the `abs_inti.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.
--	---

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"
});
```

Miscellaneous variables

Variable	Description
<code>response_size</code>	Maximum size in MB of the data fetched by external calls to ABS REST APIs. The default value is 100 MB.
<code>enable_ssl</code>	When <code>true</code> , SSL communication is enabled between ASE and ABS, and for external systems making rest API calls to ABS. See Configure SSL on page 10 for more information.

Parent topic: [AI Engine training](#)

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>/v3/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.

Parent topic: [AI Engine training](#)

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.

Parent topic: [AI Engine training](#)

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>

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 [Appendix B: 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.



Note: Make sure that you are in [user](#) mode before changing the threshold manually.

Change threshold value Tn only

The Tn threshold value can be changed for each attack type for a specific API. The initial Tx value is automatically calculated based on the gap between the values of Tn and Tx. This gap is determined at the end of the [training period](#). The minimum gap is 1, and the value of Tx always bigger than Tn. Here is an example:

Values at end of training period:

- Tn = 12
- Tx = 16
- Gap = 4 (Tx-Tn)

Threshold API is used to set Tn=13 for an API variable.

- Tx = 17 (Gap value of 4 is automatically added to new Tn value)

This difference between the value of Tn and Tx is maintained when only Tn is moved. However, the difference between the value of Tn and Tx can be changed when only Tx is changed.



Note: The value of Tn can never be more than the value of Tx.

Changing Threshold Value Tx only

Change the 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 [Appendix C](#).

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>

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 (t_n) 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 t_n , t_x , or both at the same time. If you do not wish to change the value for t_n or t_x in user mode, leave the field blank by putting "" in the Threshold API body. In the following snippet, the value of t_n and t_x 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"
}
```

Parent topic:[AI Engine training](#)

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](#) 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.

Parent topic:[AI Engine training](#)

API discovery

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

- [Discover the APIs](#)
- [Reporting the discovered APIs](#)
- [Discovery Subpaths](#)
- [ABS Discovery API](#)
- [Enable and disable discovery and update discovery interval](#)

Discover the APIs

API Behavioral Security works in tandem with ASE to automatically discover APIs in your ecosystem. The discovery process works as follows:

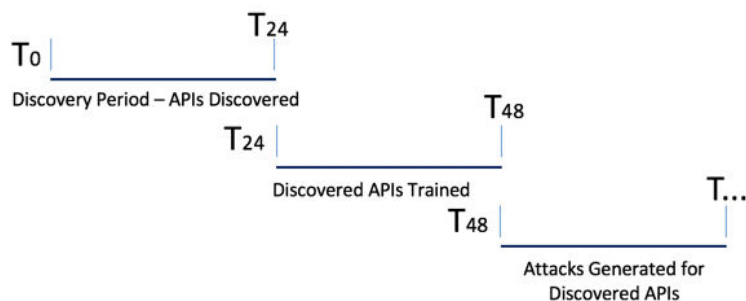
1. ASE is configured with an API JSON file with `url` as `"/"` and `hostname` as `"*"`. ASE captures the API traffic metadata in log files and forwards API traffic to backend servers. It periodically sends the log files to ABS.
2. ABS processes the ASE log files and looks for new APIs. During the discovery 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 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 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 after which APIs are discovered in your API ecosystem.
<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 Discovery Subpaths for more information.
<code>url_limit</code>	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.



Parent topic:[API discovery](#)

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.

Parent topic:[API discovery](#)

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"
});
```

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.

Access script help 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.

Parent topic: [API discovery](#)

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

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"
}
}
},
"/atmapp/upload": {
"post": {
"produces": [
"text/html"
],
"responses": {
"200": {
"description": "ok"
}
}
}
},
"/atmapp/login": {
"post": {
"produces": [
"application/json",
"text/plain"
],
"responses": {
"200": {
"description": "unauthorized"
},
"401": {
"description": "unauthorized"
}
}
}
},
"/atmapp/update": {
"put": {
"produces": [
"text/html"
],
"responses": {
"200": {
"description": "ok"
}
}
}
} } } } }

```



Note: If ASE is deployed in sideband mode, then backend host field in the output shows the IP address as `not available: 0`. The backend server field shows the IP address as `0.0.0.0`. For more information on ASE sideband mode, see the ASE Admin Guide.

- [Discover OAuth token APIs](#)

Parent topic:[API discovery](#)

Discover OAuth token APIs

ABS discovers OAuth2 token APIs when the following conditions are met:

- - An API JSON file must be created in ASE with a URL configured as / and hostname as *
 - `oauth2_access_token` must be set to `true` in the API JSON file.

During the discovery period, a sufficient volume of traffic must be sent before the API is discovered. After the API is discovered, AAD creates an API JSON file with `oauth2_access_token` enabled and adds it to ASE.

Parent topic:[ABS Discovery API](#)

Enable and disable discovery and update discovery interval

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 enabled.

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.



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 help of the script 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.

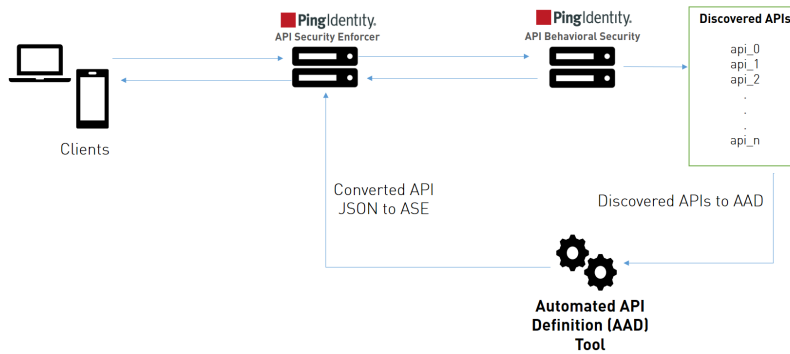
Parent topic:[API discovery](#)

Automated API Definition (AAD)

Automated API Definition (AAD) is a Java-based tool that adds ABS discovered APIs to ASE. 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.

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_blocking` is set to `true` by default. You can edit these values in ASE.

- [Install AAD](#)
- [Obfuscate keys and passwords](#)
- [Configure AAD](#)
- [Start AAD](#)
- [Stop AAD](#)
- [AAD Conversion Status](#)
- [Purge AAD log files](#)

Install AAD

Download the AAD tool from the [download](#) site. Oracle Java 8 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-3.2.1.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.

Parent topic: [Automated API Definition \(AAD\)](#)

Obfuscate keys and passwords

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:

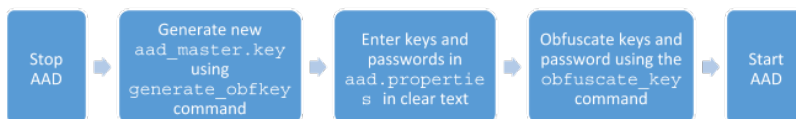
- `ase.access_key`
- `ase.secret_key`
- `abs.access_key`
- `abs.secret_key`
- `gateway.management.password`

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](#).

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.

```

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.

Parent topic: [Automated API Definition \(AAD\)](#)

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.mode</code>	Set the value to <code>discovery</code> when ASE is deployed in inline mode. 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. 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 PingIntelligence for APIs – PingAccess Integration guide. For more information on ASE modes, see the ASE Admin Guide .
<code>abs.host</code>	Hostname or IPv4 IP address of the ABS host machine.
<code>abs.port</code>	Port number of the ABS service.
<code>abs.access_key</code>	The access key of ABS. Default value is <code>abs_ak</code>
<code>abs.secret_key</code>	The secret key of ABS. Default value is <code>abs_sk</code>
<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 admin
<code>abs.query.interval</code>	The polling interval in minutes to poll for any newly discovered APIs. The default value is 10 minutes.
<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>	URL of the API Gateway. Only valid when <code>aad.mode</code> is gateway
<code>gateway.management.username</code>	Username to connect to the API Gateway. Only valid when <code>aad.mode</code> is gateway
<code>gateway.management.password</code>	Password to connect to the API Gateway. Only valid when <code>aad.mode</code> is gateway

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=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:Rs7NPeYGCU0Zku7TANJbwE12rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
# ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwE12rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
### 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==:0vOu0XUVpbvV4AaSmv5mZ1lw3WpAsj1oPF3d5Et170Y=
# 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: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=:Az82rlUffftMGpmx7une1JZUucX191102QgKvHD36vU=

```

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": [
]
}

```

```
}
}
```

Parent topic:[Automated API Definition \(AAD\)](#)

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
```

Parent topic:[Automated API Definition \(AAD\)](#)

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.
```

Parent topic:[Automated API Definition \(AAD\)](#)

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	Output to ASE
<pre>{ "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", "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": [</pre>	<pre>{ "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",</pre>

```

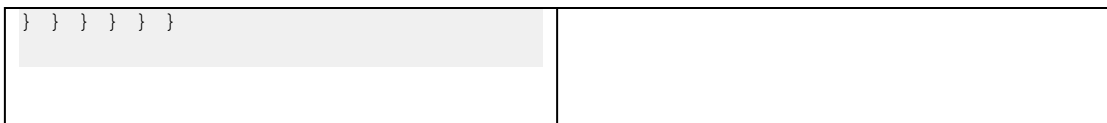
"text/html"
],
"responses": {
  "200": {
    "description": "ok"
  } } },
"/pubatmapp/login": {
  "post": {
    "produces": [
      "application/json"
    ],
    "responses": {
      "200": {
        "description": "ok"
      } } },
"/pubatmapp/zipcode": {
  "post": {
    "produces": [
      "application/json"
    ],
    "responses": {
      "200": {
        "description": "ok"
      } } },
  "get": {
    "produces": [
      "application/json"
    ],
    "responses": {
      "200": {
        "description": "ok"
      } } },
  "delete": {
    "produces": [
      "application/json"
    ],
    "responses": {
      "200": {
        "description": "ok"
      } } },
  "put": {
    "produces": [
      "application/json"
    ],
    "responses": {
      "200": {
        "description": "ok"
      } } },
"/pubatmapp/upload": {
  "post": {
    "produces": [
      "text/html"
    ],
    "responses": {
      "200": {
        "description": "ok"
      }
    }
  }
}

```

```

"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": "",
  "decoy_subpaths": []
}
}

```



Parent topic: [Automated API Definition \(AAD\)](#)

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.

Parent topic: [Automated API Definition \(AAD\)](#)

Manage REST API attack detection

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 cookie identifier, token, or IP address. An environment with multiple APIs can support a mixture of identifier types in a single ABS system. Client identifier use cases include:

- **API JSON with OAuth2 token parameter** – When an API JSON 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.
- **API JSON with cookie parameter** – 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.

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. For more information on configuring API JSON files, see [API Security Enforcer Admin Guide](#).

- [REST API attack types](#)
- [REST API attacks detected on cookie](#)
- [REST API attacks detected on token](#)
- [REST API attacks detected on IP address](#)

REST API attack types

API Behavioral Security (ABS) reports on REST API attacks using two different API calls:

- Per API attacks
- Across API attacks

Per API attacks: These attack types are reported on a specific API in your ecosystem. These attacks are based on OAuth token, cookie or an IP address. Each attack type is assigned a type ID and can be accessed using `attack` REST API of ABS. 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>/v3/abs/attack?later_date=yyyy-mm-ddT hh:mm&later_date=yyyy-mm-ddT hh:mm&api=<api_name>&type=<type_id>.

For example, https://192.168.11.166:8080/v3/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




Across API attacks: These attacks are detected across APIs in your API ecosystem. For example, if you have five APIs in your ecosystem and there is a misbehaving token, cookie or an IP, then the across API attack sniffs through all the APIs and reports the attacks.

Use the following ABS REST API to access different attack types: https://<ABS_IP:port>/v3/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/v3/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

Attack Type	Type ID
Stolen Token Attack Type 2	16
Stolen Cookie Attack Type 2	17
API Probing Replay Attack Type 2 (Cookie)	18
API Probing Replay Attack Type 2 (Token)	19
API Probing Replay Attack Type 2 (IP)	20
Excessive Client Connections (Cookie)  Note: Applicable only for Inline ASE deployment	22
Excessive Client Connections (Token)  Note: Applicable only for Inline ASE deployment	23
Excessive Client Connections (IP)  Note: Applicable only for Inline ASE deployment	24
Content Scraping Type 1 (Cookie)	25
Content Scraping Type 1 (Token)	26
Content Scraping Type 1 (IP)	27

Single Client Login Attack Type 2	30
-----------------------------------	----

Parent topic: [Manage REST API attack detection](#)

REST API attacks detected on cookie

ABS machine learning algorithms categorize attacks which are assigned a unique `type_id`. The following table lists the attacks detected based on cookie for a per API attack and across 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

* *Applicable only for Inline ASE deployment*

Parent topic: [Manage REST API attack detection](#)

REST API attacks detected on token

The following table lists the REST API attacks using tokens detected by ABS. The attacks can be on a single API or across 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	4	Single API

	to attack an API service.		
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
-------------------------	---	----	------------

* *Applicable only for Inline ASE deployment*

Parent topic: [Manage REST API attack detection](#)

REST API attacks detected on IP address

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

* *Applicable only for Inline ASE deployment*

Parent topic: [Manage REST API attack detection](#)

Manage 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.

- [Automatic blocking of attacks with ASE](#)
- [Whitelist and blacklist management](#)
- [Per API blocking](#)

Automatic blocking of attacks with ASE

Automatic blocking of attacks with ASE

When enabled in API Security Enforcer, ABS detected attack lists (OAuth2 token, IP addresses and/or cookies suspected of executing attacks) are automatically sent to ASE nodes which terminate current sessions and block future access for clients 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
```

Parent topic: [Manage Attack blocking](#)

Whitelist and blacklist management

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](#)
- [Manage Blacklist](#)
- [Blacklist to whitelist transition](#)

Parent topic: [Manage Attack blocking](#)

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
```

Parent topic: [Whitelist and blacklist management](#)

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 ad233edqsd1d23redwefew
cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
```

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

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

- **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.

Parent topic:[Whitelist and blacklist management](#)

Blacklist to whitelist transition

When you delete a black list entry which was created by ABS or ASE, it is automatically added to the whitelist and no longer blocked by ASE. However, CLI added entries deleted from the blacklist are not added to the whitelist. When the blacklist is cleared, list entries are not transitioned to the whitelist.

Parent topic:[Whitelist and blacklist management](#)

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}`

Parent topic:[Manage Attack blocking](#)

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>/v3/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. Few examples of API output is produced where [the cookie is obfuscated](#). For the Admin user, the cookie, token or the API key is not obfuscated.

- [Consolidated result of attack types](#)
- [API Deception](#)
- [Real-time Detected attacks for inline ASE](#)
- [Anomalous activity reporting](#)

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>/v3/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/v3/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>.

Example: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)"
            }
        ]
    }
]
}
]

```

Parent topic:[Attack reporting](#)

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/v3/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"
    }
  ]
}
```

```

"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.

Parent topic: [Attack reporting](#)

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/v3/abs/attack?
later_date=2018-07-16&earlier_date=2018-07-16&api=atmapp&type=101](https://192.168.11.138:8080/v3/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": []
}

```

Parent topic: [Attack reporting](#)

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"
]
}
]
}
],
}

```

Parent topic:[Attack reporting](#)

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/v3/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

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 provides in-depth information on the activities performed by a client using an OAuth2 token, IP or cookie across APIs. The client identifier is included in attack reports and can be used to generate a forensic report which provides detailed information on the activity of an individual token, IP or cookie.



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": []
}
}

```

List hacker's URL

The List Invalid URLs 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"
  ]
}
}
}
}

```

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
  }
]
```



```

{
  "company": "ping identity",
  "name": "api_metrics",
  "description": "This report contains a summary and detailed api key
  metrics across all APIs",
  "earlier_date": "Tue Feb 13 18:00:00:000 2018",
  "later_date": "Sun Feb 18 18:00:00:000 2018",
  "api_name": "atm_app_public",
  "req_resp_summary": {
    "api_url": "/atm_app_public",
    "total_requests": 2508,
    "success": 2246,
    "sessions": 2,
    "no_sessions": 1,
    "most_popular_method": "POST",
    "most_popular_device": "UNKNOWN",
    "most_popular_ips": [
      "127.0.0.1",
      "3.1.1.4"
    ],
    "servers": [
      {
        "server": "127.0.0.1:3000",
        "count": 2507
      }
    ],
  },
  "req_resp_details": {
    "api_url": "/atm_app_public",
    "session_details": [
      {
        "session_id": "session_protocol",
        "total_requests": 1,
        "source_ip": [
          {
            "ip": "127.0.0.1",
            "count": 1,
            "method": [
              "GET"
            ]
          }
        ],
      },
    ],
    "user_agent": [
      {
        "user_agent": "DOWNLOAD",
        "count": 1
      }
    ],
    "path_info": [
      {
        "path": "/atm_app_public/get400",
        "count": 1
      }
    ],
  },
}

```

```

"device": [
  {
    "device": "UNKNOWN",
    "count": 1
  }
],
"server": []
},
{
  "session_id": "session11",
  "total_requests": 2506,
  "source_ip": [
    {
      "ip": "127.0.0.1",
      "count": 2506,
      "method": [
        "DELETE",
        "POST",
        "PUT",
        "GET"
      ]
    }
  ],
  "user_agent": [
    {
      "user_agent": "DOWNLOAD",
      "count": 2506
    }
  ],
  "path_info": [
    {
      "path": "/atm_app_public/post400",
      "count": 218
    },
    {
      "path": "/atm_app_public/put400",
      "count": 18
    },
    {
      "path": "/atm_app_public/delete200",
      "count": 208
    },
    {
      "path": "/atm_app_public/get400",
      "count": 14
    },
    {
      "path": "/atm_app_public/put200",
      "count": 152
    },
    {
      "path": "/atm_app_public/delete400",
      "count": 10
    }
  ],

```



```
{
  "path": "/atm_app_public/get200",
  "count": 104
},
{
  "path": "/atm_app_public/post200",
  "count": 1782
}
],
"device": [
  {
    "device": "UNKNOWN",
    "count": 2506
  }
],
"server": [
  {
    "server": "127.0.0.1:3000",
    "count": 2506
  }
]
],
"no_session": {
  "request_details": [
    {
      "total_requests": 1,
      "source_ip": [
        {
          "ip": "3.1.1.4",
          "count": 1,
          "method": [
            "GET"
          ]
        }
      ]
    }
  ],
  "user_agent": [
    {
      "user_agent": "DOWNLOAD",
      "count": 1
    }
  ],
  "path": "/atm_app_public/get400",
  "device": [
    {
      "device": "UNKNOWN",
      "count": 1
    }
  ],
  "server": [
    {
      "server": "127.0.0.1:3000",
      "count": 1
    }
  ]
}
```



```

"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
    }
  ]
}

```

Backend errors

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 for Google Chrome browser
- 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.

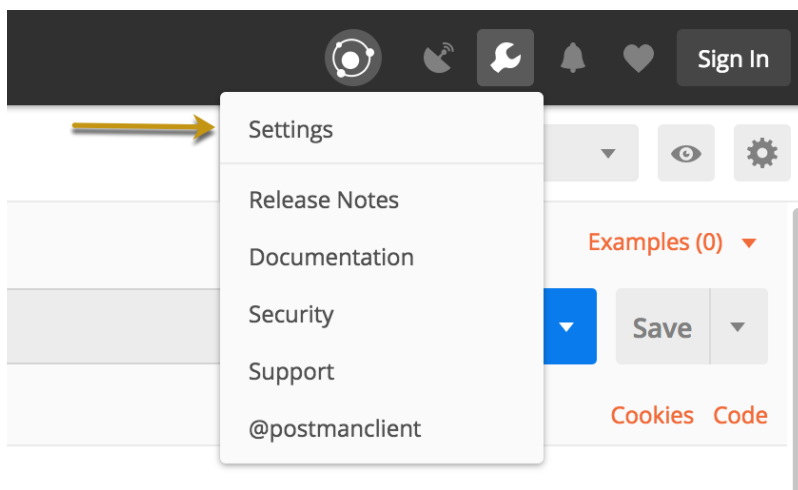
- [ABS self-signed certificate with Postman](#)
- [View ABS reports in Postman](#)

ABS self-signed certificate with Postman

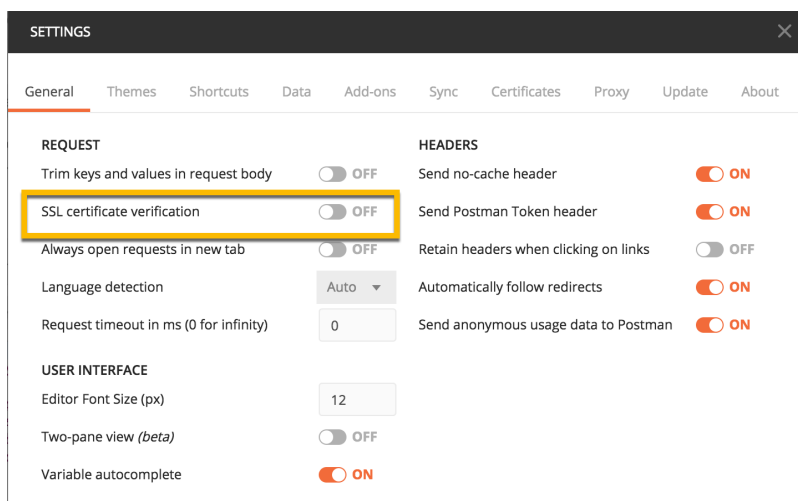
Using ABS self-signed certificate with Postman

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:

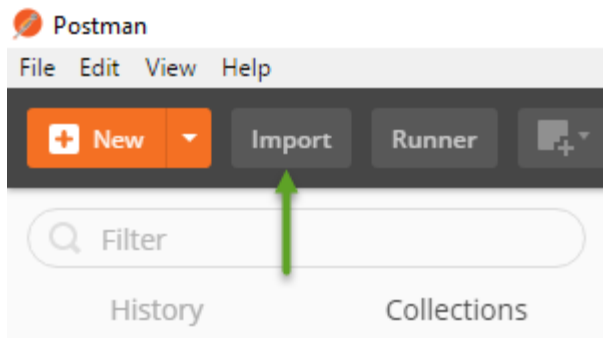



Parent topic: [API reports using Postman](#)

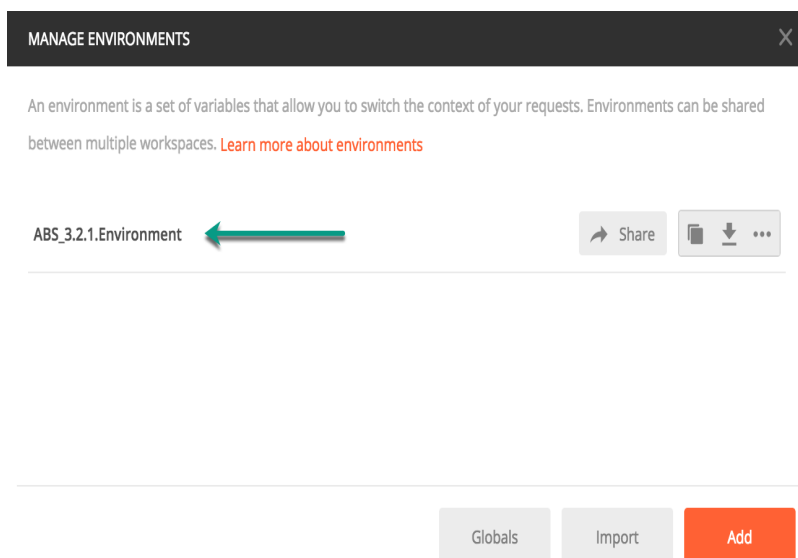
View ABS reports in Postman

To view the reports, complete the following steps:

1. Download `ABS_3.2.1_Environment` and `ABS_3.2.1_Reports` JSON files from **API Reports Using Postman** folder on Ping Identity [Download](#) site. These configuration files will be used by Postman.
2. [Download](#) and install the Postman application 6.2.5 or higher.
3. In Postman, **import** the two Ping Identity files downloaded in step 1 by clicking the Import button.



4. After importing the files, click the gear  button in the upper right corner.
5. In the **MANAGE ENVIRONMENTS** pop-up window, click **ABS_3.2.1_Environment**



6. In the pop-up window, configure the following values and then click **Update**
 - **Server:** IP address of the ABS node for which the `dashboard_node` was set to `true` in the [abs.properties](#) file.
 - **Port:** Port number of the ABS node.
 - **Access_Key_Header** and **Secret_Key_Header:** Use the Admin user or Restricted user header. A Restricted user sees obfuscated value of OAuth token, cookie and API keys. For more information of different types of user, see [ABS users for API reports](#)
 - **Access_Key** and **Secret_Key:** The Access Key and Secret Key configured in the `opt/pingidentity/mongo/abs_init.js` for either admin or restricted user. Make sure that access key and secret key corresponds to the admin or restricted user header configured.
 - **API_Name:** The name of the API for which you want to generate the reports.
 - **Later_Date:** A date which is more recent in time. For example, if the query range is between March 12 and March 14, then the later date would be March 14.
 - **Earlier_Date:** A date which is past in time. For example, if the query range is between March 12 and March 14, then the earlier date would be March 12.



Note: Do not edit any fields that start with the word `system`.

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	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	PingIntelligenceApp	PingIntelligenceApp	
<input checked="" type="checkbox"/>	Later_Date	2019-03-31T18:00	2019-03-31T18:00	
<input checked="" type="checkbox"/>	Earlier_Date	2019-01-31T13:30	2019-01-31T13:30	
<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}}/apikey?later_date={{Later_Date}}&earl...	
<input checked="" type="checkbox"/>	System_OAuth_Toke...	{{System_URL}}/oaut...	{{System_URL}}/oauth?later_date={{Later_Date}}...	

ⓘ 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.

Parent topic: [API reports using Postman](#)

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](#)
- [Anomalies API](#)
- [OAuth2 Token Forensics](#)
- [IP Forensics API](#)
- [Cookie 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
- Discovery REST API
- Decoy REST API
- GET Threshold REST API
- PUT Threshold REST API
- Metrics REST API
- API Key Metrics REST API
- OAuth2 Token Metrics REST API
- Anomalies REST API
- OAuth2 Token Forensics REST API
- IP Forensics REST API
- Cookie Forensics REST API
- Attack Types REST APIs
- Flow Control REST API
- Blocked Connection REST API
- Backend Error REST API
- List Valid URLs REST API
- List Hacker's URL REST 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: </v3/abs/admin>

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
  "company": "ping identity",
  "name": "api_admin",
  "description": "This report contains status information on all APIs, ABS"
```

```

clusters, and ASE logs",
  "across_api_prediction_mode": false,
  "api_discovery": {
    "status": false
  },
  "apis": [
    {
      "api_name": "apikeyquery",
      "host_name": "*",
      "url": "/apikeyquery",
      "api_type": "decoy-incontext",
      "creation_date": "Wed Feb 06 19:36:22 IST 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": "",
      "token": false,
      "training_started_at": "training not started yet",
      "training_duration": "24 hours",
      "prediction_mode": false
    },
    {
      "api_name": "apikeyheader",
      "host_name": "*",
      "url": "/apikeyheader",
      "api_type": "decoy-incontext",
      "creation_date": "Wed Feb 06 19:36:22 IST 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": "",
      "token": false,
      "training_started_at": "training not started yet",
      "training_duration": "24 hours",
      "prediction_mode": false
    },
    {
      "api_name": "atmapp",
      "host_name": "*",
      "url": "/atmapp",
      "api_type": "decoy-incontext",
      "creation_date": "Wed Feb 06 19:36:22 IST 2019",
      "servers": 4,
      "protocol": "https",
      "cookie": "",
      "token": false,
      "training_started_at": "training not started yet",
      "training_duration": "24 hours",
      "prediction_mode": false
    },
    {
      "api_name": "pubatmapp",
      "host_name": "*",
      "url": "/pubatmapp",
      "api_type": "decoy-incontext",
      "creation_date": "Wed Feb 06 19:36:22 IST 2019",

```

```

        "servers": 4,
        "protocol": "https",
        "cookie": "JSESSIONID",
        "token": false,
        "training_started_at": "training not started yet",
        "training_duration": "24 hours",
        "prediction_mode": false
    }
],
"abs_cluster": {
    "abs_nodes": [
        {
            "node_ip": "192.168.11.165",
            "os": "Red Hat Enterprise Linux Server release 7.4 (Maipo)",
            "cpu": "24",
            "memory": "62G",
            "filesystem": "76%",
            "bootup_date": "Tue Feb 05 16:12:41 IST 2019"
        }
    ],
    "mongodb_nodes": [
        {
            "node_ip": "192.168.11.168",
            "status": "up"
        }
    ]
},
"ase_logs": [
    {
        "ase_node": "13eea2fc-64d0-4c51-b663-b1093b0bf7a5",
        "last_connected": "Wed Feb 06 19:41:07 IST 2019",
        "logs": {
            "start_time": "Wed Feb 06 19:36:26 IST 2019",
            "end_time": "Wed Feb 06 19:41:07 IST 2019",
            "gzip_size": "27.51MB"
        }
    }
],
"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"
  }
}
}

```

Parent topic: [ABS external REST APIs](#)

Discovery REST API

Description: The Discovery API discovers all the APIs that are available in your API ecosystem.

Method: GET

URL: </v3/abs/discovery>

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "name": "api_discovery_summary",
  "description": "This report contains summary of discovered APIs",
  "summary": [
    {
      "api_name": "api_0",
      "host": "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"
},
}

```

Parent topic: [ABS external REST APIs](#)

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: `/v3/abs/decoy?later_date<>&earlier_date<>`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "name": "decoy_api_metrics",
  "description": "This report contains detailed information on client access
to each decoy API",
  "earlier_date": "Tue Jan 11 17:50:00:000 2018",
  "later_date": "Tue Jan 11 18:00:00:000 2018",
  "api_name": "atmapp",
  "api_type": "decoy-incontext",
  "decoy_url": [
    "/atmapp/decoy"
  ],
  "summary": [
    {
      "decoy_url": "/atmapp/decoy",
      "unique_ip_count": 122,
      "total_requests": 240,
      "most_used_methods": {
        "GET": 88,
        "DELETE": 32,
        "ABDU": 32,
        "POST": 30,
        "PUT": 26
      },
      "most_used_ips": {
        "100.64.9.37": 4,
        "100.64.10.79": 4,

```

```

"100.64.31.183": 2,
"100.64.20.213": 2,
"100.64.34.239": 2
},
"most_used_devices": {
"UBUNTU": 76,
"MAC_OS_X": 69,
"WINDOWS_7": 61,
"WINDOWS_XP": 34
},
"most_used_content_types": {
"UNKNOWN": 184,
"multipart/form-data": 56
}
}
],
"details": [
{
"decoy_url": "/atmapp/decoy",
"source_ip": [
{
"ip": "100.64.31.183",
"total_requests": 2,
"method_count": {
"GET": {
"count": 2
}
},
"url_count": {
"/atmapp/decoy": 2
}
},
{
"ip": "100.64.14.28",
"total_requests": 2,
"method_count": {
"POST": {
"count": 2,
"payload_characteristics": {
"multipart/form-data": [
"354 bytes"
]
}
}
}
},
"url_count": {
"/atmapp/decoy": 2
}
},
{
"ip": "100.64.0.55",
"total_requests": 2,
"method_count": {
"GET": {

```

```

"count": 2
}
},
"url_count": {
"/atmapp/decoy": 2
}
},
{
"ip": "100.64.20.152",
"total_requests": 2,
"method_count": {
"DELETE": {
"count": 2
}
},
"url_count": {
"/atmapp/decoy": 2
}
}
]
}
]
}

```

Parent topic: [ABS external REST APIs](#)

GET Threshold REST API

Description: The GET Threshold API fetches the threshold values for attack types.

Method: GET

URL for an API: [/v3/abs/attack/threshold?api=<api_name>](#)

URL for across API: [/v3/abs/attack/threshold?id=<type_id>](#). The API name is not specified in the URL for fetching the threshold value. Type ID is the [attack ID](#)

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response for an API

```

{
  "company": "ping identity",
  "name": "api_threshold",
  "description": "This report contains threshold settings for 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"
      }
    }
  }
]
}

```

Parent topic:[ABS external REST APIs](#)

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: [/v3/abs/attack/threshold](#)

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Input for an API

```

{
  "api_name" : "atmapp",
  "mode": "system",
  "ioc_threshold": [
    {
      "type": "api_memory_post",
      "variable": "A",
    },
    {

```



```

"type": "api_memory_put",
"variable": "B"
}
]
}

```

The following is the response when the threshold values are set:

```

{
  "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"
}

```

Parent topic: [ABS external REST APIs](#)

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: /v3/abs/metrics?later_date=<>&earlier_date=<>api=<api_name>

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
  "company": "ping identity",
  "name": "api_metrics",
  "description": " This report contains metrics for request/response traffic
  for the specified API",
  "earlier_date": "Mon Jan 13 18:00:00:000 2018",
  "later_date": "Wed Jan 15 18:00:00:000 2018",
  "api_name": "shop",
  "req_resp_summary": {
    "api_url": "shopapi",
    "total_requests": 342102,
    "success": 279360,
    "sessions": 0,
    "no_sessions": 342102,
    "most_popular_method": "GET",
    "most_popular_device": "MAC_OS_X",
    "most_popular_ips": [
      "10.10.1.38",
      "10.10.1.39",
      "10.10.1.37"
    ]
  },
  "servers": [
    {
      "server": "192.168.11.164:3001",
      "count": 5357
    },
    {
      "server": "192.168.11.164:3002",
      "count": 5354
    },
    {
      "server": "192.168.11.164:3003",
      "count": 5358
    },
    {
      "server": "192.168.11.164:3004",
      "count": 1667
    }
  ],
  "req_resp_details": {
    "api_url": "shopapi",
    "session_details": [],

```

```
"no_session": {
  "request_details": [
    {
      "total_requests": 14865,
      "source_ip": [
        {
          "ip": "10.10.1.24",
          "count": 152,
          "method": [
            "POST"
          ]
        },
        {
          "ip": "10.10.1.71",
          "count": 482,
          "method": [
            "PUT"
          ]
        }
      ],
      "user_agent": [
        {
          "user_agent": "SAFARI",
          "count": 7187
        },
        {
          "user_agent": "FIREFOX",
          "count": 12536
        },
        {
          "user_agent": "MOZILLA",
          "count": 5509
        },
        {
          "user_agent": "CHROME",
          "count": 29241
        }
      ],
      "server": [
        {
          "server": "192.168.11.164:3001",
          "count": 723
        },
        {
          "server": "192.168.11.164:3002",
          "count": 689
        },
        {
          "server": "192.168.11.164:3003",
          "count": 749
        },
        {
          "server": "192.168.11.164:3004",
          "count": 237
        }
      ]
    }
  ]
}
```


API Key Metrics REST API

Description: The API Key-based Metrics API is used to fetch the metrics for API Keys across all APIs.

Method: GET

URL: /v3/abs/apikeys?later_date=<yy-mm-dd>T<hh:mm>&earlier_date==<yy-mm-dd>T<hh:mm>

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
  "company": "ping identity",
  "name": "api_key_metrics",
  "description": "This report contains a summary and detailed api key
  metrics across all APIs",
  "earlier_date": "Fri Jan 19 13:00:00:000 2018",
  "later_date": "Sat Jan 20 18:00:00:000 2018",
  "summary": {
    "api_keys": 325,
    "total_requests": 329
  },
  "details": [
    {
      "api_key": "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": "NW00DLM68PFQ3XTL",
```

```

"total_requests": 1,
"ip_list": [
  {
    "ip": "100.64.20.62",
    "total_requests": 1,
    "devices": {
      "WINDOWS_XP": 1
    },
    "methods": {
      "DELETE": 1
    },
    "urls": {
      "/apikeyheader/zipcode": 1
    },
    "apis": {
      "apikeyheader": 1
    }
  }
],
{
  "api_key": "86ELLUSN6RAHEPF7",
  "total_requests": 1,
  "ip_list": [
    {
      "ip": "100.64.17.79",
      "total_requests": 1,
      "devices": {
        "MAC_OS_X": 1
      },
      "methods": {
        "GET": 1
      },
      "urls": {
        "/apikeyheader/zipcode": 1
      },
      "apis": {
        "apikeyheader": 1
      }
    }
  ],
  {
    "api_key": "5JSKZZ53TGBQZ8V2",
    "total_requests": 1,
    "ip_list": [
      {
        "ip": "100.64.33.183",
        "total_requests": 1,
        "devices": {
          "WINDOWS_7": 1
        },
        "methods": {
          "POST": 1
        }
      }
    ]
  }
]

```

```

},
"urls": {
"/apikeyheader/login": 1
},
},
"apis": {
"apikeyheader": 1
}
}
]
}
]
}
}

```

Parent topic: [ABS external REST APIs](#)

OAuth2 Token Metrics REST API

Description: The OAuth2 token-based API is used to fetch the metrics for OAuth2 token across all APIs.

Method: GET

URL: [/v3/abs/oauthtokens?later_date=<yy-mm-dd>T<hh:mm>&earlier_date==<yy-mm-dd>T<hh:mm>](#)

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "name": "oauth_token_metrics",
  "description": "This report contains a summary and detailed oauth token metrics across all APIs",
  "earlier_date": "Tue Feb 13 18:00:00:000 2018",
  "later_date": "Sun Feb 18 18:00:00:000 2018",
  "summary": {
    "tokens": 30,
    "total_requests": 163250
  },
  "details": [
    {
      "token": "token_highresptime",
      "total_requests": 2,
      "ip_list": [
        {
          "ip": "127.0.0.1",
          "total_requests": 2,
          "devices": {
            "UNKNOWN": 2
          }
        }
      ]
    }
  ]
}

```

```

},
"methods": {
"GET": 2
},
"urls": {
"/2_atm_app_oauth/longresponse": 1,
"/atm_app_oauth/longresponse": 1
},
"apis": {
"atm_app_oauth": 1,
"2_atm_app_oauth": 1
}
}
],
},
{
"token": "token10",
"total_requests": 4596,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 4596,
"devices": {
"UNKNOWN": 4596
},
"methods": {
"DELETE": 148,
"POST": 1036,
"GET": 1796,
"PUT": 1616
},
"urls": {
"/2_atm_app_oauth/put200": 656,
"/atm_app_oauth/delete200": 68,
"/2_atm_app_oauth/put400": 152,
"/atm_app_oauth/delete400": 6
},
"apis": {
"atm_app_oauth": 2298,
"2_atm_app_oauth": 2298
}
}
],
},
{
"token": "token14",
"total_requests": 7604,
"ip_list": [
{
"ip": "127.0.0.1",
"total_requests": 7604,
"devices": {
"UNKNOWN": 7604
},

```



```

"methods": {
  "DELETE": 1596,
  "POST": 160,
  "GET": 4000,
  "PUT": 1848
},
"urls": {
  "/2_atm_app_oauth/put200": 846,
  "/atm_app_oauth/delete200": 742,
  "/2_atm_app_oauth/put400": 78,
  "/2_atm_app_oauth/get400": 264
},
"apis": {
  "atm_app_oauth": 3802,
  "2_atm_app_oauth": 3802
}
]
},
{
  "token": "token_type2memory",
  "total_requests": 2,
  "ip_list": [
    {
      "ip": "127.0.0.1",
      "total_requests": 2,
      "devices": {
        "UNKNOWN": 2
      },
      "methods": {
        "POST": 2
      },
      "urls": {
        "/2_atm_app_oauth/post200": 1,
        "/atm_app_oauth/post200": 1
      },
      "apis": {
        "atm_app_oauth": 1,
        "2_atm_app_oauth": 1
      }
    }
  ],
  "token": "token_method",
  "total_requests": 2,
  "ip_list": [
    {
      "ip": "127.0.0.1",
      "total_requests": 2,
      "devices": {
        "UNKNOWN": 2
      },
      "methods": {

```

```

"HEAD": 2
},
"urls": {
"/2_atm_app_oauth/get400": 1,
"/atm_app_oauth/get400": 1
},
"apis": {
"atm_app_oauth": 1,
"2_atm_app_oauth": 1
}
}
]
}
]
}
}

```

Parent topic: [ABS external REST APIs](#)

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: `/v3/abs/anomalies?later_date=<>earlier_date=<>&api=<api_name>`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "name": "api_anomalies",
  "description": "This report contains information on anomalous activity
  on the specified API.",
  "earlier_date": "Sun Jan 12 18:00:00:000 2018",
  "later_date": "Tue Jan 14 18:00:00:000 2018",
  "api_name": "shop",
  "anomalies_summary": {
    "api_url": "shopapi",
    "total_anomalies": 14,
    "most_suspicious_ips": [],
    "most_suspicious_anomalies_urls": []
  },
  "anomalies_details": {
    "url_anomalies": {
      "suspicious_sessions": [],
      "suspicious_requests": []
    }
  }
}

```

```

},
"ioc_anomalies": [
{
"anomaly_type": "API Memory Attack Type 2",
"cookies": [
{
"cookie": "AMAT_2_H",
"access_time": [
"Mon Jan 13 01:01:33:589 2018"
]
},
{
"cookie": "AMAT_2_H",
"access_time": [
"Mon Jan 13 01:01:33:589 2018"
]
}
]
},
{
"anomaly_type": "Data Exfiltration Attack",
"cookies": [
{
"cookie": "data_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": [

```


URL: /v3/abs?

later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&token=<oauth2_token>

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
  "company": "ping identity",
  "name": "api_abs_token",
  "description": "This report contains a summary and detailed information on
  metrics, attacks and anomalies for the specified token across all APIs.",
  "earlier_date": "Tue Feb 13 18:00:00:000 2018",
  "later_date": "Sun Feb 18 18:00:00:000 2018",
  "summary": {
    "total_requests": 6556,
    "total_attacks": 2,
    "total_anomalies": 0
  },
  "details": {
    "metrics": {
      "token": "token1",
      "total_requests": 6556,
      "ip_list": [
        {
          "ip": "127.0.0.1",
          "total_requests": 6556,
          "devices": {
            "UNKNOWN": 6556
          }
        }
      ],
      "methods": {
        "DELETE": 472,
        "POST": 140,
        "GET": 1944,
        "PUT": 4000
      }
    },
    "urls": {
      "/atm_app_oauth/delete200": 218,
      "/atm_app_oauth/get200": 850,
      "/atm_app_oauth/post400": 8,
      "/atm_app_oauth/post200": 62,
      "/atm_app_oauth/put400": 62,
      "/atm_app_oauth/get400": 122,
      "/atm_app_oauth/put200": 1938,
      "/atm_app_oauth/delete400": 18,
      "/2_atm_app_oauth/put200": 1938,
      "/2_atm_app_oauth/post200": 62,
      "/2_atm_app_oauth/delete200": 218,
      "/2_atm_app_oauth/delete400": 18,
    }
  }
}
```

```

"/2_atm_app_oauth/put400": 62,
"/2_atm_app_oauth/post400": 8,
"/2_atm_app_oauth/get400": 122,
"/2_atm_app_oauth/get200": 850
},
"apis": {
  "atm_app_oauth": 3278,
  "2_atm_app_oauth": 3278
}
]
},
"attack_types": {
  "API Memory Attack Type 1": [
    "atm_app_oauth",
    "2_atm_app_oauth"
  ],
  "Data Poisoning Attack": [
    "atm_app_oauth",
    "2_atm_app_oauth"
  ]
},
"anomaly_types": {}
}
}

```

Parent topic: [ABS external REST APIs](#)

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: `/v3/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&IP=<IP_address>`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "name": "api_abs_ip",
  "description": " This report contains a summary and detailed information
  on all attacks, metrics, and anomalies for the specified IP address on
  the defined API.",
  "summary": {
    "total_requests": 18222,

```

```

"total_ioctypes": 0,
"total_anomalies": 0
},
"details": {
  "ioc_types": [],
  "metrics": {
    "no_session": [
      {
        "start_time": "Sat Jan 04 15:30:00:000 2018",
        "end_time": "Sat Jan 04 15:39:59:952 2018",
        "total_requests": 2749,
        "source_ip": "100.64.10.203",
        "path": "/atmapp/login"
        "methods": [
          "GET"
        ]
      },
      {
        "start_time": "Sat Jan 04 15:30:00:000 2018",
        "end_time": "Sat Jan 04 15:39:59:952 2018",
        "total_requests": 2952,
        "source_ip": "100.64.10.203",
        "path": "/atmapp/upload"
      },
      {
        "start_time": "Sat Jan 04 15:30:00:000 2018",
        "end_time": "Sat Jan 04 15:39:59:952 2018",
        "total_requests": 9547,
        "source_ip": "100.64.10.203",
        "path": "/atmapp/zipcode"
      },
      {
        "start_time": "Sat Jan 04 15:30:00:000 2018",
        "end_time": "Sat Jan 04 15:39:59:952 2018",
        "total_requests": 2964,
        "source_ip": "100.64.10.203",
        "path": "/atmapp/update"
      }
    ],
    "session": [
      {
        "session_id": "ZP7FE32357SPVT5X",
        "start_time": "Sat Jan 04 15:35:14:241 2018",
        "end_time": "Sat Jan 04 15:35:14:241 2018",
        "total_requests": 1,
        "source_ip": [
          {
            "ip": "100.64.10.203",
            "count": 1,
            "method": [
              "POST"
            ]
          }
        ]
      }
    ]
  }
}

```

```

"user_agent": [
  {
    "user_agent": "IE11",
    "count": 1
  }
],
"path_info": [
  {
    "path": "/atmapp/upload",
    "count": 1
  }
],
"device": [
  {
    "device": "WINDOWS_7",
    "count": 1
  }
],
},
{
  "device": [
    {
      "device": "MAC_OS_X",
      "count": 1
    }
  ]
},
},
"start_time": "Sat Jan 04 15:40:00:000 2018",
"end_time": "Sat Jan 04 15:30:00:000 2018",
"api_name": "atmapp"
}

```

Parent topic: [ABS external REST APIs](#)

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: [/v3/abs?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&cookie=<cookie_value>](#)

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response


```

{
  "company": "ping identity",
  "name": "api_abs_cookie",
  "description": "This report contains a summary and detailed information
    on all attacks, metrics, and anomalies for the specified cookie on
    the defined API",
  "earlier_date": "Mon Jan 17 06:40:00:000 2018",
  "later_date": "Mon Jan 17 07:00:00:000 2018",
  "api_name": "shop",
  "summary": {
    "total_requests": 501,
    "total_anomalies": 0,
    "total_ioc": 3
  },
  "details": {
    "ioc_types": [
      "data_exfiltration_attack",
      "cookie_dos_attack",
      "extreme_client_activity_attack"
    ],
    "metrics": [
      {
        "session_id": "extreme_client_activity_500_request",
        "start_time": "Mon Jan 17 06:47:19:687 2018",
        "end_time": "Mon Jan 17 06:47:20:505 2018",
        "total_requests": 501,
        "source_ip": [
          {
            "ip": "100.100.10.12",
            "count": 501,
            "method": [
              "POST",
              "GET"
            ]
          }
        ],
        "user_agent": [
          {
            "user_agent": "CHROME",
            "count": 501
          }
        ],
        "path_info": [
          {
            "path": "/shopapi/get",
            "count": 500
          },
          {
            "path": "/shopapi/login",
            "count": 1
          }
        ],
        "device": [
          {

```

```

"device": "LINUX",
"count": 501
}
]
}
],
"anomalies": []
}
}

```

Parent topic: [ABS external REST APIs](#)

Attack Types REST 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 1-13 and 50-53.

Method: GET

URL: `/v3/abs/attack?later_date<>&earlier_date<>&api=<api_name>&type=<type_id>`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "description": " Client (IP or Cookie) extracting an abnormal amount of
  data for given API",
  "earlier_date": "Wed Jan 01 08:20:00:000 2018",
  "later_date": "Sun Jan 05 13:20:00:000 2018",
  "api_name": "atmapp",
  "ioc_type": "Data Exfiltration",
  "ips": [
    {
      "ip": "100.64.6.50",
      "access_time": [
        "Sat Jan 04 16:09:59:935 2018"
      ]
    },
    {
      "ip": "100.64.6.51",
      "access_time": [
        "Sat Jan 04 16:09:59:935 2018",
        "Sat Jan 04 16:39:59:996 2018"
      ]
    }
  ]
}

```

Parent topic: [ABS external REST APIs](#)

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: `/v3/abs/flowcontrol?later_date=<>&earlier_date=<>&api=<api_name>`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
  "company": "ping identity",
  "name": "api_flowcontrol",
  "description": "This report contains flow control information for the
    specified API.",
  "earlier_date": "Wed Jan 01 08:20:00:000 2018",
  "later_date": "Sun Jan 05 13:20:00:000 2018",
  "api_name": "websocket",
  "summary": {
    "client_spike": 610,
    "connection_queued": 0,
    "connection_quota_exceeded": 0,
    "bytes_in_spike": 2743,
    "bytes_out_spike": 287
  },
  "details": {
    "client_spike": [],
    "server_spike": [
      {
        "request_time": "Fri Jan 09 17:19:55:977 2016",
        "connection_id": "147378243",
        "source_ip": "100.64.26.163",
        "destination_api": "/atmapp/login"
      },
      {
        "request_time": "Fri Jan 09 17:19:55:991 2016",
        "connection_id": "1919058221",
        "source_ip": "100.64.20.230",
        "destination_api": "/atmapp/zipcode"
      }
    ]
  }
}
```

```

"connections_queued": [],
"connections_rejected": [],
"bytes_in_spike": [],
"bytes_out_spike": []
}
}

```

Parent topic: [ABS external REST APIs](#)

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 `/v3/abs/bc?later_date=<>T<hh:mm>&earlier_date=<>T<hh:mm>&details=true`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "earlier_date": "Wed Jan 01 08:20:00:000 2018",
  "later_date": "Sun Jan 05 13:20:00:000 2018",
  "api_blocked_connections": [
    {
      "date": "05September2016",
      "blocked_connections": [
        {
          "apiproxy_node": "204101a4-8b70-489d-98e9-aa3f6e67a93f",
          "blocked_connections": [
            {
              "category": "ioc",
              "details": []
            },
            {
              "category": "api",
              "details": [
                {
                  "source": "100.64.31.235",
                  "type": "no_backend_available",
                  "destination_api": "/atmapp/zipcode"
                },
                {
                  "source": "100.64.25.184",
                  "type": "no_backend_available",
                  "destination_api": "/atmapp/zipcode"
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}

```

```

},
{
  "source": "100.64.6.137",
  "type": "no_backend_available",
  "destination_api": "/atmapp/zipcode"
},
{
  "source": "100.64.1.251",
  "type": "no_backend_available",
  "destination_api": "/atmapp/zipcode"
}
]
}
]
}
]
}
]
}
]
}
}

```

Parent topic: [ABS external REST APIs](#)

Backend Error REST API

Description: The Backend Error API displays errors reported by the backend servers.

Method: GET

URL: `/v3/abs/be?earlier_date=<>T<hh:mm>&later_date=<>T<hh:mm>&api=<api_name>`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```

{
  "company": "ping identity",
  "name": "api_backend_errors",
  "description": "This report contains details of backend error
  codes for the specified API",
  "earlier_date": "Wed Jan 01 08:20:00:000 2018",
  "later_date": "Sun Jan 05 13:20:00:000 2018",
  "api_name": "atmapp",
  "backend_error_summary": [
    {
      "error_code": "403",
      "error": "Forbidden",
      "count": 0
    },
    {

```

```
"error_code": "404",
"error": "Not Found",
"count": 0
},
{
"error_code": "500",
"error": "Internal Server Error",
"count": 16
},
{
"error_code": "503",
"error": "Service Unavailable",
"count": 0
},
{
"error_code": "504",
"error": "Gateway Timeout",
"count": 0
}
],
"backend_error_details": [
{
"error_code": "403",
"details": []
},
{
"error_code": "404",
"details": []
},
{
"error_code": "500",
"details": [
{
"server": "192.168.11.164:3001",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.5.183:24078",
"request_cookie": ""
},
{
"server": "192.168.11.164:3002",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.18.126:61932",
"request_cookie": ""
},
{
"server": "192.168.11.164:3004",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.27.176:2908",
"request_cookie": "JSESSIONID=6UQANJWB42U4A4PF"
},
{
"server": "192.168.11.164:3004",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.14.237:21973",
```

```

"request_cookie": "JSESSIONID=LJ66P3NQW5SDVW8Q"
},
{
"server": "192.168.11.164:3003",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.5.101:5523",
"request_cookie": ""
},
{
"server": "192.168.11.164:3003",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.23.132:14473",
"request_cookie": "JSESSIONID=NCTZ4RSOZP2IT2OU"
},
{
"server": "192.168.11.164:3003",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.5.197:50811",
"request_cookie": ""
},
{
"server": "192.168.11.164:3003",
"request_url": "/atmapp/zipcode",
"request_ip": "100.64.26.70:49425",
"request_cookie": ""
}
]
},
{
"error_code": "503",
"details": []
},
{
"error_code": "504",
"details": []
}
]
}

```

Parent topic: [ABS external REST APIs](#)

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: [/v3/abs/validurl?api=<api_name>](#)

	Header	Value
Access Key	x-abs-ak	<string>

Secret Key	x-abs-sk	<string>
-------------------	----------	----------

Sample Response

```
{
  "company": "ping identity",
  "name": "api_url_list",
  "description": "This report provides information on access to each
  unique URL for the specified API",
  "api_name": "shop",
  "host_name": "app",
  "api_url": "shopapi",
  "allowed_methods": [
    "GET",
    "PUT",
    "POST",
    "DELETE",
    "HEAD"
  ],
  "url_list": [
    {
      "protocol": "HTTP/1.1",
      "urls": [
        {
          "url": "/shopapi/get_delay",
          "total_count": 11,
          "methods": [
            {
              "method": "GET",
              "count": 11
            }
          ]
        },
        {
          "url": "/shopapi/post",
          "total_count": 62109,
          "methods": [
            {
              "method": "POST",
              "count": 62109
            }
          ]
        }
      ],
    },
    {
      "url": "/shopapi/get_mb",
      "total_count": 2,
      "methods": [
        {
          "method": "GET",
          "count": 2
        }
      ]
    }
  ]
}
```



```
},
{
  "url": "/shopapi/login",
  "total_count": 2686,
  "methods": [
    {
      "method": "POST",
      "count": 2686
    }
  ]
},
{
  "url": "/shopapi/get?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
    }
  ]
}
] ] }
```

Parent topic: [ABS external REST APIs](#)

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: `/v3/abs/hackersurl?api=<api_name>&earlier_date=""&later_date=""`

	Header	Value
Access Key	x-abs-ak	<string>
Secret Key	x-abs-sk	<string>

Sample Response

```
{
  "company": "ping identity",
  "description": "This report contains list of hackers URL for given API",
  "name": "api_hackers_url",
  "api_name": "universal_api",
  "invalid_urls": [
    {
      "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,-- ",
      "ips": [
        "127.0.0.1"
      ]
    },
    {
      "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL,NULL,NULL#",
      "ips": [
        "127.0.0.1"
      ]
    },
    {
      "url": "/index.php?id=(SELECT 46 FROM(SELECT COUNT(*),CONCAT(0x717a71,))",
      "ips": [
        "127.0.0.1"
      ]
    },
    {
      "url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL,NULL#",
      "ips": [
        "127.0.0.1"
      ]
    }
  ],
}
```

```

{
"url": "/index.php?id=abc UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,#",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc' UNION ALL SELECT NULL,NULL,NULL,NULL,,NULL#",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc UNION ALL SELECT NULL,NULL,NULL,NULL,NULL#",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL-- ",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc') UNION ALL SELECT NULL,NULL-- ",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc UNION ALL SELECT NULL,NULL,NULL,NULL,NULL#",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc%' UNION ALL SELECT NULL-- ",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc) UNION ALL SELECT NULL,NULL,NULL,NULL-- ",
"ips": [
"127.0.0.1"
]
},
{
"url": "/index.php?id=abc' UNION ALL SELECT NULL,NULL,NULL-- ",
"ips": [
"127.0.0.1"
]
}
}

```

```
]
}
```

Parent topic: [ABS external REST APIs](#)

Threshold range for Tn and Tx

Threshold range for Tn and Tx

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

Attack Type	type_id	Variable A (Range)	Variable B (Range)	Variable C (Range)
REST API				
Data Exfiltration	1	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]
Single Client Login	2	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA
Multi Client Login	3	Tn = [1-100] Tx = "na"	NA	NA
Stolen Cookie / Access Token	4	Tn = [2-10]	Tn = [1-19], Tx = [2-20]	Tn = [2-10]
API Memory Attack Type 1	5	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]
API Memory Attack Type 2	6	Tn = [1-32] Tx = [2-33]	Tn = [1-19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]
Cookie DoS	7	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA
API Probing Replay	8	Tn = [1-99] Tx = [2-100]	NA	NA
API DoS Attack Type 1	9	Tn = [1-100] Tx = "[2-100]"	NA	NA
Extreme Client Activity	10	Tn = [1-19] Tx = [2-20]	NA	NA
Extreme App Activity	11	Tn = [1-19] Tx = [2-20]	NA	NA
API DoS Attack	12	Tn = [1- 100] Tx = "na"	NA	NA
API DDoS Attack Type 2	13	NA	NA	NA

Data Deletion	14	Tn = [1- 19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	NA
Data Poisoning	15	Tn = [1- 19] Tx = [2-20]	Tn = [1-99] Tx = [2-100]	Tn = [1-32] Tx = [2-33]
Stolen Token Attack Type 2	16	Tn = [2-10] Tx = "na"	Tn = [1-100]	Tn = [1-100]
Stolen Cookie Attack Type 2	17	Tn = [2-10] Tx = "na"	Tn = [1-100]	Tn = [1-100]
API Probing Replay Attack 2 (client identifier: cookie)	18	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA
API Probing Replay Attack 2 (client identifier: token)	19	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA
API Probing Replay Attack 2 (client identifier: IP address)	20	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA
Data Exfiltration Attack Type 2	21	Tn = [1-42] Tx = [2-43]	Tn = [0-30]	Tn = [1-100]
Excessive Client Connections (client identifier : cookie)	22	Tn = [1-19], Tx =[2-20]	NA	NA
Excessive Client Connections (client identifier : token)	23	Tn = [1-19], Tx =[2-20]	NA	NA
Excessive Client Connections (client identifier : IP address)	24	Tn = [1-19], Tx =[2-20]	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] This attack has another variable D. The threshold range is Tn = [1-19] Tx = [2-20]

Content Scraping Type 1 (client identifier : token)	26	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20] This attack has another variable D. The threshold range is Tn = [1-19] Tx = [2-20]
Content Scraping Type 1 (client identifier : IP address)	27	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20] This attack has another variable D. The threshold range is Tn = [1-19] Tx = [2-20]
Content Scraping Type 2	28	Tn = [1-29] Tx = [2-30]	Tn = [1-100]	NA
Unauthorized client attack (client identifier : IP address)	29	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	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
WebSocket API				
WS Cookie Attack	50	Tn = [1-99] Tx = [2-100]	Tn = [1-19] Tx = [2-20]	NA
WS Identity Attack	51	Tn = [1-19] Tx = [2-20]	Tn = [1-19] Tx = [2-20]	NA
WS DoS Attack	53	Tn = [1- 100] Tx = "na"	NA	NA
WS Data Exfiltration Attack	54	Tn = [1- 100] Tx = "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.

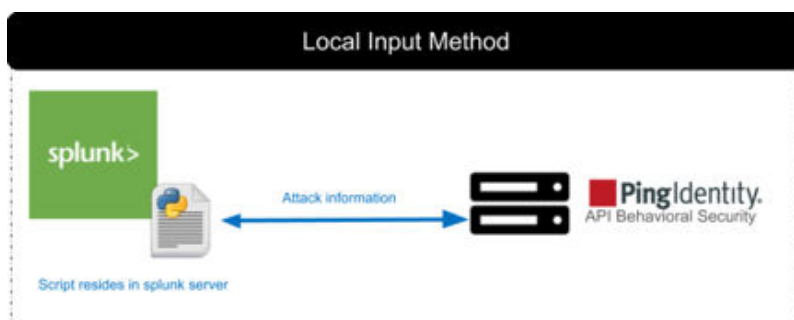
Installing and configuring Splunk for PingIntelligence is a two step process.

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

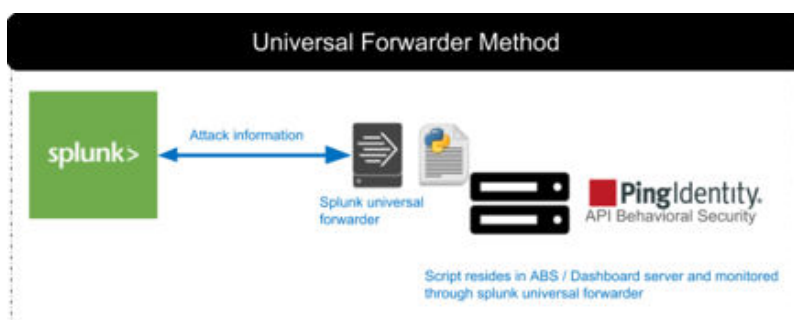
After you have configured the method to send data, the PingIntelligence ABS script creates an individual event based on the attacks reported by ABS and posts it to splunk.

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](#)
- [Install and configure Splunk for PingIntelligence](#)
- [Types of data captured](#)
- [Local Input Method installation and configuration](#)
- [Splunk Universal Forwarder method installation and configuration](#)

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.

Parent topic: [Splunk for PingIntelligence](#)

Install and configure Splunk for PingIntelligence

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:

1. Configure a new source type by navigating to **Splunk Enterprise** → **Settings** → **Source Types** → **New Source type**. The source type events page is displayed.
2. Configure the **New Source type**. The fields are defined in the following table:

Name	Value
Source type name	pi_events_source_type
Destination app	Search and Reporting (Can change for your apps)
Category	Structures
Indexed Extractions	json
Timestamp	%a %b %d %H:%M:%S %Y
BREAK_ONLY_BEFORE	(\{)
MUST_BREAK_AFTER	(\})

Edit Source Type: pi_events_source_type ×

Description:

Destination app:

Category:

Indexed Extractions:

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	×
pulldown_type	true	×

[New setting](#)

Parent topic: [Splunk for PingIntelligence](#)

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 access_type: attack
api_name: shop
attack_type: Data Exfiltration Attack Type 2
identifier: IbxvhvPyih
identifier_count: 1
identifier_type: cookie
source_ip: [ [-]
100.10.5.81
]
timestamp: Mon Mar 18 05:40:00:000 2019
}
Show as raw text
host = ip-172-31-19-163.ap-south-1.compute.internal | source = /opt/pingidentity/splunk/data/pi_events.data | sourcetype = pi_events_source_type

```

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 = /opt/pingidentity/splunk/data/pi_events.data | sourcetype = pi_events_source_type
```

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


```

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	<ul style="list-style-type: none"> • 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
<code>abs.host</code>	The hostname or IPv4 address of ABS host
<code>abs.port</code>	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>	<p>The time in past for which the script window fetches data. The value is specified in seconds. Recommended value is 1800 seconds.</p> <p>Example: 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.</p>

abs.query.window	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.
logfile	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**

Note: The interval is set to 10-minutes (600-seconds) as shown 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.

Select New

pi_events_source_type

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](#)

App Context Apps Browser (appsbrowser)

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](#)

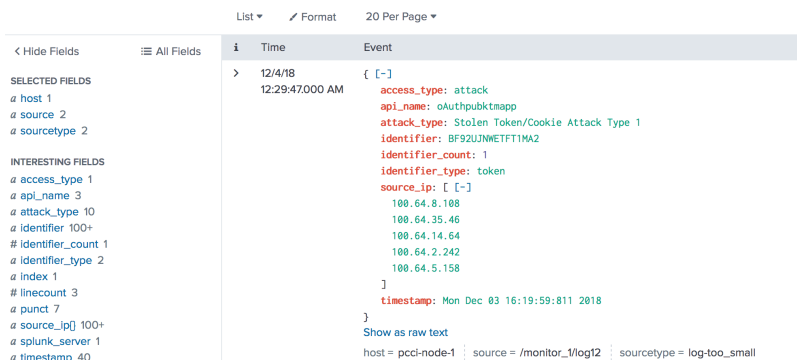
Host field value ABS HOST

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 configuration without impacting production indexes. You can always change this setting later. [Learn More](#)

Index pi_events Create a new index

- **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.



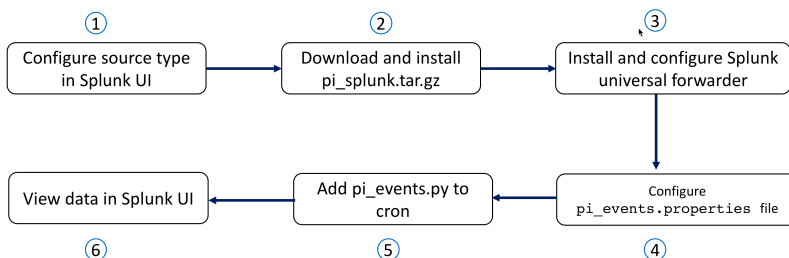
Parent topic: [Splunk for PingIntelligence](#)

Stopping the script

You can stop the script for local input method by navigating to **Settings ->Data Inputs ->Scripts (Local Inputs) ->Delete option**

Splunk Universal Forwarder method installation and configuration

The installation and configuration process of Splunk universal forwarder method is depicted in the diagram below:



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-3.2.1.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 sources and forward that data into Splunk software for indexing and consolidation. They can scale to tens of thousands of remote systems, collecting terabytes of data.

Choose Your Installation Package

Windows
 Linux
 Solaris
 Mac OS
 FreeBSD
 AIX

64-bit	2.6+ kernel Linux distributions	.rpm	23.53 MB	Download Now
		.tgz	23.47 MB	Download Now
		.deb	17.15 MB	Download Now

- 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.

<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. Example: 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 recommended value is 600-seconds.
<code>logfile</code>	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

	i	Time	Event
	>	14/03/2019	{ [-]
		06:03:07.000	<pre> access_type: attack api_name: shop-books attack_type: Data Exfiltration Attack Type 1 identifier: 188.18.6.6 identifier_count: 1 identifier_type: ip source_ip: [[-] 188.18.6.6] timestamp: Wed Mar 13 07:27:44:566 2019 </pre>
			Show as raw text
			host = ABS:HOST source = /opt/splunk/bin/scripts/pi_events.py sourcetype = pi_events_source_type
	>	14/03/2019	{ [-]
		06:03:07.000	<pre> access_type: attack api_name: shop-books attack_type: Data Exfiltration Attack Type 1 identifier: 188.18.9.39 identifier_count: 1 identifier_type: ip source_ip: [[-]] timestamp: Wed Mar 13 07:09:59:932 2019 </pre>



Note: If no data is available in Splunk, check your firewall settings.

Parent topic: [Splunk for PingIntelligence](#)

Stopping the script

You can stop the script for universal forwarder method by completing the following steps:

1. Open crontab

```
# crontab -e
```

2. Stop monitor of data recovery

```
# ./splunk remove monitor /opt/pingidentity/splunk/data/
Removed monitor of '/opt/pingidentity/splunk/data/'.
```

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.

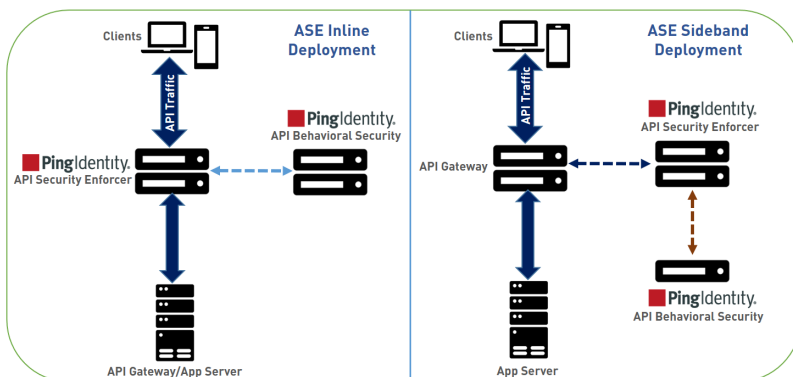
PingIntelligence for APIs detects anomalous behavior on APIs, as well as the data and applications exposed via APIs, and can automatically block attacks across your API environment. For example, attempts to bypass login systems using botnet credential stuffing attacks or stolen tokens are recognized 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.

Introduction to API Security Enforcer

ASE supports multiple deployments modes to provide customers flexibility in deploying PingIntelligence for APIs API cybersecurity. This ASE admin guide covers the following deployment modes:

Inline ASE - ASE receives API client traffic and then routes the traffic to a backend API gateway or directly to App Servers. ASE applies real time security and passes API metadata to the ABS Engine for AI powered advanced attack detection. ABS engine notifies ASE of attacks, and ASE then blocks the rogue clients.

Sideband ASE – An API gateway receives API client traffic and then makes API calls to pass API metadata to ASE for processing. ASE passes the API metadata to the ABS Engine for AI powered advanced attack detection. ABS engine notifies ASE of attacks, and ASE then works with API gateway to block inbound rogue client requests. See ASE sideband chapter for more information.



The following table shows a summary of features available in each deployment options.

Security Features	Inline	Sideband
Interface to ABS AI Engine for AI powered attack detection	Yes	Yes
API deception – decoy APIs look like legitimate APIs to hackers. After accessing a decoy API, a hacker is quarantined, plus activity information is collected.	Yes	Yes
Real-time client blocking based on lists with ASE detected attacks, ABS AI Engine detected attacks, or customer-built lists. Blocking can be based on OAuth2 tokens, API keys, cookies, and IP addresses.	Yes	Yes
Black and whitelist management of tokens, API keys, cookies, IP addresses	Yes	Yes

Real-time blocking of API clients with traffic that deviates from API attributes.	Yes	No
Dynamic mapping of public API identity to private internal API identity	Yes	No
Custom API error messages prevent disclosure of sensitive error information.	Yes	No
Admin Features		
Simple deployment with modular JSON configuration files	Yes	Yes
Live updates – Add/remove without loss of traffic or stopping services.	Yes	Yes
Obfuscation – Keys and passwords are obfuscated	Yes	Yes
Active-active clustering – Supports scaling and resiliency: all nodes are peers and self-learn the configuration, traffic information, and security updates.	Yes	Yes
Syslog information messages sent to Syslog servers in RFC 5424 format.	Yes	Yes
Automatic API discovery discovers API JSON configuration data	Yes	Yes
CLI and REST API for management and automation tool integration.	Yes	Yes
Linux PAM -based administrator authentication with existing Linux tools.	Yes	Yes
Audit log captures administrative actions for compliance reporting.	Yes	Yes

Distributed inbound flow control limits client traffic and server traffic	Yes	No
Multiprotocol Layer 7 routing and load balancing of WebSocket, REST API	Yes	No
Secure connection between ASE and ABS. Secure connection also between ASE and ASE REST APIs	Yes	Yes

Using the ASE Admin Guide

The API Security Enforcer (ASE) Admin Guide caters to two different deployment options, inline ASE and sideband ASE.

The guide is divided into the following four parts:

- **ASE Administration** – This section is applicable to both the deployment types and contains basic administrative configuration information.
- **ASE Inline** – This section is specific to ASE inline configuration and the protection features that ASE inline offers.
- **ASE Sideband** – This section is specific to ASE sideband configuration and protection features that the ASE sideband offers.
- **Appendices** – The appendices list the commands for inline and sideband, REST APIs, and audit logs.

ASE 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](#)
- [ASE interfaces](#)
- [Start and stop ASE](#)
- [Change default settings](#)
- [Obfuscate keys and passwords](#)
- [PKCS#12 keystore](#)
- [ASE directory structure](#)
- [ASE cluster setup](#)
- [Tune host system for high performance](#)
- [Customizing ASE ports](#)
- [Configure SSL for external APIs](#)
- [Configure SSL for management APIs](#)
- [Configure native and PAM authentication](#)
- [ASE management, access and audit logs](#)

- [Purge log files](#)
- [Configure syslog](#)
- [Configure email notifications](#)

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

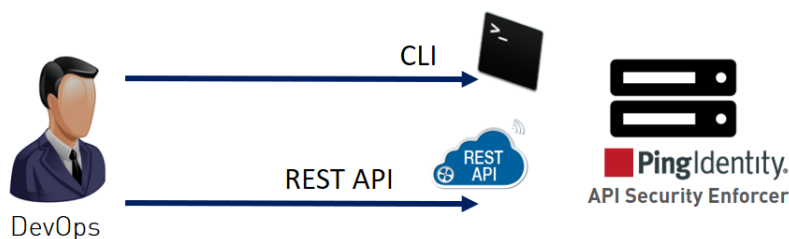
If your existing license has expired, obtain a fresh license from PingIntelligence for APIs sales team and replace the license file in the `/opt/pingidentity/ase/config` directory. Make sure to stop and start ASE after the license file is updated.

Parent topic: [ASE administration](#)

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](#) and [CLI for sideband ASE](#)

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



Note: After initial start-up, all configuration changes must be made using **cli.sh** or ASE REST APIs. This includes adding a server, deleting a server, adding a new API, and so on. After manually editing an operational JSON file, follow [Updating a Configured API](#)

CLI commands include the following:

help command

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

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

version command

To query system information, enter the version command:

```
/opt/pingidentity/ase/bin/cli.sh version
Ping Identity Inc., ASE 3.1.1
Kernel Version : 3.10
Operating System : Red Hat Enterprise Linux Server release 7.0 (Maipo)
Build Date : Fri Aug 24 13:43:22 UTC 2018
```

status command

To get ASE status, enter the status command:

```
/opt/pingidentity/ase/bin/cli.sh status
Ping Identity Inc., API Security Enforcer
status : started
http/ws : port 80
https/wss : port 443
firewall : enabled
abs : disabled, ssl: enabled
abs attack : disabled
audit : enabled
ase detected attack : disabled
attack list memory : configured 128.00 MB, used 25.60 MB, free 102.40 MB
```

ASE REST API

The ASE REST API is used to administer ASE or integrate ASE with third-party products. Using the ASE REST API, you can configure ASE and display ASE statistics, including the number of backend servers, the number of APIs, and so on.

ASE REST API commands consist of the following:

- **API:** Create API (POST), Read API (GET), List API (GET), Update API (PUT), Delete API (DELETE)
- **Server:** Create Server (POST), Read Server (GET), Delete Server (DELETE)
- **Session:** Read Persistent Connections (GET)
- **Cluster:** Read Cluster (GET)
- **Firewall:** Read Firewall Status (GET), Update Firewall Status (POST)
- **Flow Control:** Read flow control (GET), Update flow control for API (POST), Update flow control of a Server for an API (POST)

Parent topic: [ASE administration](#)

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

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

```
/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 3.2.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Stop ASE

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

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

Parent topic: [ASE administration](#)

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 3.2.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Change keystore password

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

Command: update_keystore_password

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

Change admin password

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

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

Parent topic: [ASE administration](#)

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:

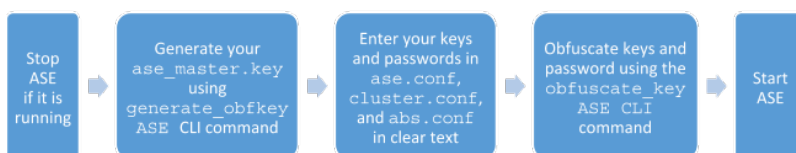
- `ase.conf` – Email and keystore (PKCS#12) password
- `cluster.conf` – ABS access and secret key
- `abs.conf` – Cluster authentication key

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.



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

Obfuscate keys and passwords

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

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

Please take a backup of `config/ase_master.key`, `config/ase.conf`, `config/abs.conf`, and `config/cluster.conf` before proceeding

If config keys and passwords are already obfuscated using the current master key, they are not obfuscated again

Following keys will be obfuscated:

`config/ase.conf`: `sender_password`, `keystore_password`

`config/abs.conf`: `access_key`, `secret_key`

`config/cluster.conf`: `cluster_secret_key`

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

obfuscating `config/ase.conf`, success

obfuscating `config/abs.conf`, success

obfuscating `config/cluster.conf`, success

Start ASE after keys and passwords are obfuscated.



Important: After the keys and passwords are obfuscated, the `ase_master.key` must be moved to a secure location from ASE for security reasons. If you want to restart ASE, the `ase_master.key` must be present in the `/opt/pingidentity/ase/config/` directory.

Parent topic: [ASE administration](#)

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
```

Parent topic: [ASE administration](#)

ASE directory structure

During the installation process, ASE creates the following directories:

Directory Name	Purpose
<code>config</code>	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.
<code>data</code>	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. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  Note: The scripts in the <code>bin</code> directory are not editable. </div>
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"> • <code>check_ports.sh</code> Check ABS ports • <code>open_ports_ase.sh</code>: Run this script on the ASE machine to open the default ASE ports: 80, 443, 8010, and 8020. • Purge logs

Parent topic: [ASE administration](#)

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
- Realtime 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 a secured channel.

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

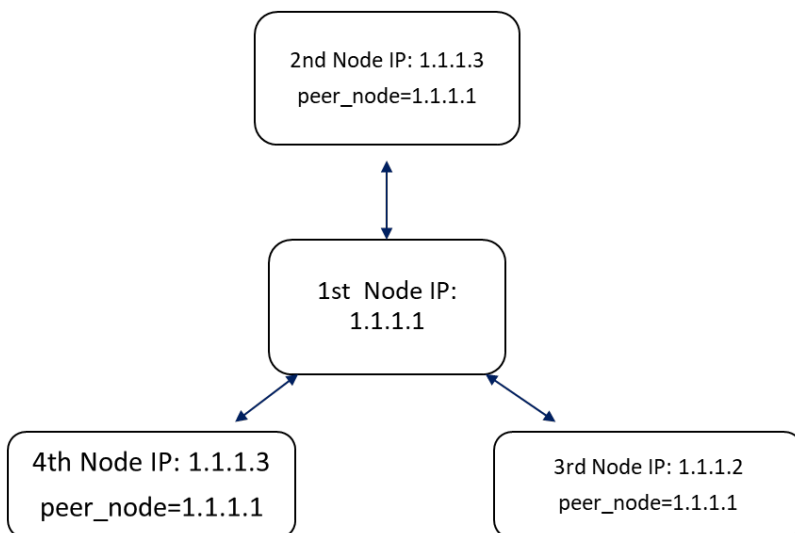
ASE cluster is a distributed node architecture. Ping Identity recommends that one cluster node be designated the management node through which all configuration changes are performed. This helps maintain consistency of operations across nodes. However, no restrictions exist on using other nodes in the 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.

- [Start ASE cluster](#)
- [Scale up the ASE cluster](#)
- [Scale down ASE cluster](#)
- [Delete ASE cluster node](#)
- [Stop ASE cluster](#)

Parent topic: [ASE administration](#)

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:7LFcqXQlqkW9kldQoFg0nJoLSojnzHDbD3i

```

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>
```

Parent topic: [ASE cluster setup](#)

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
```

Parent topic:[ASE cluster setup](#)

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

Parent topic:[ASE cluster setup](#)

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:

Parent topic:[ASE cluster setup](#)

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.

Parent topic:[ASE cluster setup](#)

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. It is not necessary to run this script for ASE to work.



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

The following commands are for tuning RHEL 7. For tuning Ubuntu 16 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
```

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
```

To secure your API environment, 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.



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

Parent topic: [ASE administration](#)

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.

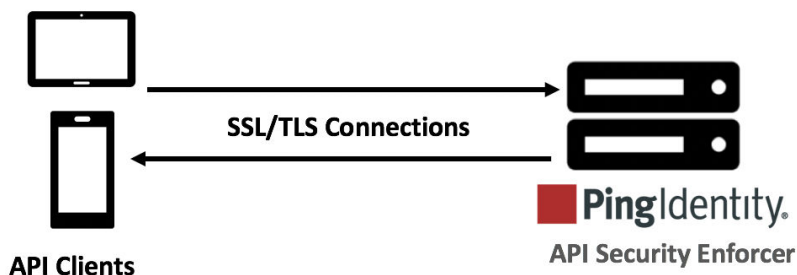
Parent topic: [ASE administration](#)

Configure SSL for external APIs

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

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

The steps provided in this section are for certificate and key generated for connections between the client and ASE as depicted in the illustration below:



In a cluster setup:

1. Stop all the ASE cluster nodes
2. Configure the certificate on the [management node](#)
3. Start the cluster nodes one by one for the certificates to synchronize across the nodes

Method 1: Use CA-signed certificate

To use Certificate Authority (CA) signed SSL certificates, follow the process to create a private key, generate a Certificate Signing Request (CSR), and request a certificate as shown below:



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

To use a CA-signed certificate:

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


```

/opt/pingidentity/ase/bin/cli.sh create_key_pair -u admin -p
Warning: create_key_pair will delete any existing key_pair, CSR and self-signed certificate
Do you want to proceed [y/n]:y
Ok, creating new key pair. Creating DH parameter may take around 20 minutes. Please wait
Key created in keystore
dh param file created at /opt/pingidentity/ase/config/certs/dataplane/dh1024.pem
      
```
2. Create a CSR. ASE takes you through a CLI-based interactive session to create a CSR.


```

/opt/pingidentity/ase/bin/cli.sh create_csr -u admin -p
Warning: create_csr will delete any existing CSR and self-signed certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >US
State > Colorado
Location >Denver
Organization >Pingidentity
Organization Unit >Pingintelligence
Common Name >ase
Generating CSR. Please wait...
OK, csr created at /opt/pingidentity/ase/config/certs/dataplane/ase.csr
      
```
3. Upload the CSR created in step 2 to the CA signing authority's website to get a CA signed certificate.
4. Download the CA-signed certificate from the CA signing authority's website.
5. Use the CLI to import the signed CA certificate into ASE. The certificate is imported into the keystore.


```

/opt/pingidentity/ase/bin/cli.sh import_cert <CA signed certificate path> -u admin -p
Warning: import_cert will overwrite any existing signed certificate
Do you want to proceed [y/n]:y
Exporting certificate to API Security Enforcer...
OK, signed certificate added to keystore
      
```
6. Restart ASE by first stopping and then starting ASE.

Method 2: Use self-signed certificate

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

To create a self-signed certificate

1. Create a private key. ASE CLI is used to generate a 2048-bit private key which is in the `/opt/pingidentity/ase/config/certs/dataplane/dh1024.pem` directory.

```
/opt/pingidentity/ase/bin/cli.sh create_key_pair -u admin -p
Warning: create_key_pair will delete any existing key_pair, CSR and self-
signed certificate
Do you want to proceed [y/n]:y
Ok, creating new key pair. Creating DH parameter may take around 20
minutes. Please wait
Key created in keystore
dh param file created at /opt/pingidentity/ase/config/certs/dataplane/
dh1024.pem
```

2. Create a CSR file:

```
/opt/pingidentity/ase/bin/cli.sh create_csr -u admin -p
Warning: create_csr will delete any existing CSR and self-signed
certificate
Do you want to proceed [y/n]:y
please provide following info
Country Code >US
State >colorado
Location >Denver
Organization >PI
Organization Unit >TEST
Common Name >yoursiteabc.com
Generating CSR. Please wait...
OK, csr created at /opt/pingidentity/ase/config/certs/dataplane/ase.csr
```

3. Create a self-signed certificate. Use the CLI to produce a self-signed certificate using the certificate request located in `/pingidentity/ase/config/certs/dataplane/ase.csr`

```
/opt/pingidentity/ase/bin/cli.sh create_self_sign_cert -u admin -p
Warning: create_self_sign_cert will delete any existing self-signed
certificate
Do you want to proceed [y/n]:y
Creating new self-signed certificate
OK, self-sign certificate created in keystore
```

4. Restart ASE by stopping and starting.

Method 3: Import an existing certificate and key pair

To install an existing certificate, complete the following steps and import it into ASE. If you have intermediate certificate from CA, then append the content to your server crt file.

1. Import key pair:

```
/opt/pingidentity/ase/bin/cli.sh import_key_pair private.key -u admin -p
Warning: import_key_pair will overwrite any existing certificates
```

```
Do you want to proceed [y/n]:y
Exporting key to API Security Enforcer...
OK, key pair added to keystore
```

2. Import the .crt file in ASE using the **import_cert** CLI command

```
/opt/pingidentity/ase/bin/cli.sh import_cert server-crt.crt -u admin -p
Warning: import_cert will overwrite any existing signed certificate
Do you want to proceed [y/n]:y
Exporting certificate to API Security Enforcer...
OK, signed certificate added to keystore
```

3. Restart ASE by stopping and starting.

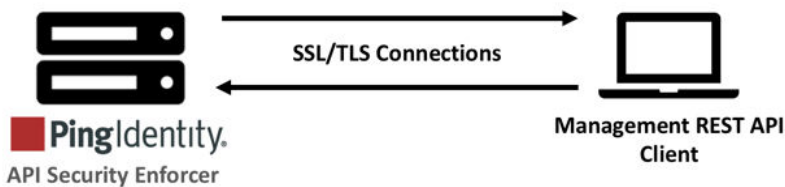
Parent topic:[ASE administration](#)

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 `/opt/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
```

- Restart ASE by stopping and starting.

Parent topic: [ASE administration](#)

Configure native and PAM authentication

ASE provides two types of authentication:

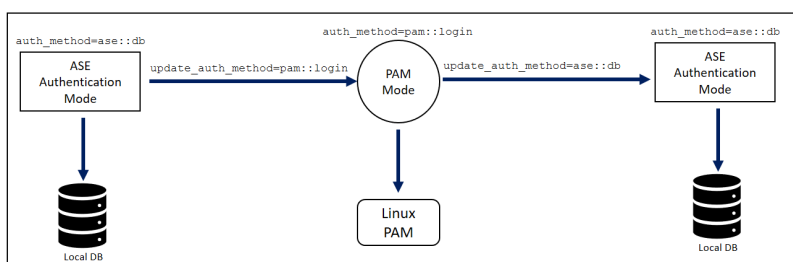
- Linux Pluggable Authentication Module (PAM)
- ASE native authentication (default method)

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.

Parent topic:[ASE administration](#)

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 (i.e. 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.

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
```

Parent topic: [ASE administration](#)

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.

Parent topic: [ASE administration](#)

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

```

Parent topic: [ASE administration](#)

Configure email notifications

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.

```
Email parameters in ase.conf are configured based on your email server.
An example configuration of email-related parameters is displayed below.
; 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.your-domain.com
smtp_port=587
sender_email=sender-mail-id@yourdomain.com
sender_password=
receiver_email=receiver-mail-id@yourdomain.com
; Defines threshold for an email alert. 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/removing an API
 - Adding/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 [Configuring 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:
 - 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

Email reports

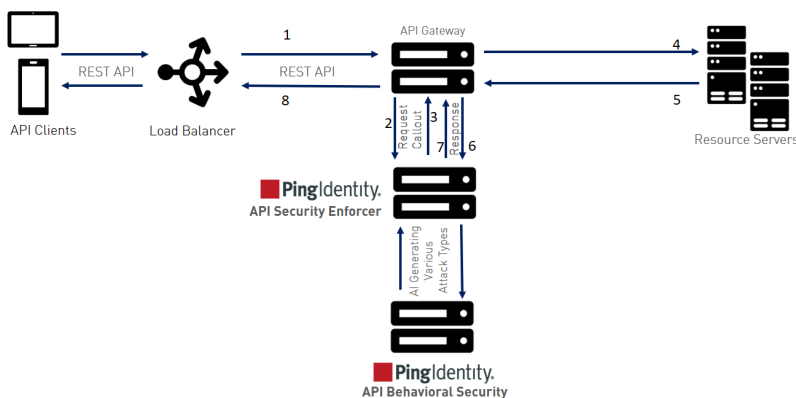
ASE sends reports at a frequency in number of days configured in `ase.conf` file. The report is sent at midnight, 00:00:00 hours based on the local system time. The report contains the following:

- 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

Parent topic: [ASE administration](#)

Sideband API Security Enforcer

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 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: 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

If you want to have a secure 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](#)
- [Activate API cybersecurity](#)
- [API deception environment](#)
- [ASE configuration for ABS AI-based security](#)
- [CLI for sideband ASE](#)


ASE configuration - ase.conf

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

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

ASE system level configuration entails modifying parameters in the `ase.conf` file located in the `config` directory. Some values have default settings which can be modified to support application requirements. The parameter values and descriptions are included in the following table:

Parameter	Description
ASE mode	

<code>mode</code>	Change the mode to sideband for ASE to work in a sideband mode. The default value is inline.
ASE ports	
<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.
<code>management_port</code>	Management port used for CLI and REST API management. The default value is 8010.
ASE administration and audit	
<code>admin_log_level</code>	The level of log detail captured. Options include: Fatal – 1, Error – 2, Warning – 3, Info – 4, Debug – 5
<code>enable_audit</code>	When set to true , ASE logs all actions performed in ASE in the audit log files. The default value is true .
<code>syslog_server</code>	Syslog server hostname or IPv4 address:port number. Leave this parameter blank for no syslog generation.
<code>hostname_refresh</code>	N/A
<code>auth_method</code>	Authentication method used for administrator access. See Configuring Native and PAM Authentication for more information on the two options. <ul style="list-style-type: none"> • <code>ase::db</code> (Default - Native authentication) • <code>pam::ldap</code> (Linux-PAM authentication with script)
<code>ase_health</code>	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. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  Note: Do not configure the /ase URL in an API JSON file. </div>
<code>enable_1G</code>	N/A
<code>http_ws_process</code>	The number of HTTP processes. It is set to 1. Do not change this value.
<code>https_wss_process</code>	The number of HTTPS or processes. It is set to 1. Do not change this value.

<code>enable_access_log</code>	When true, log client traffic request and response information. Default value is true.
<code>flush_log_immediate</code>	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.
<code>attack_list_memory</code>	The amount of memory used for maintaining black and whitelists. The default value is 128 MB.
ASE cluster	
<code>enable_cluster</code>	When true, run setup in cluster mode. The default value is false, run in standalone mode.
Security	
<code>enable_sslv3</code>	When true, enable SSLv3. Default value is false.
<code>server_ca_cert_path</code>	N/A
<code>enable_xff</code>	N/A
<code>enable_firewall</code>	When true, activates the ASE firewall. The default value is true.
Real-time API security	
<code>enable_ase_detected_attack</code>	When true, activates the real-time security in ASE. The default value is false.
API deception	
<code>decoy_alert_interval</code>	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)	
<code>enable_abs</code>	When true, send access log files to ABS for generating API metrics and detecting attacks using machine learning algorithms.
<code>enable_abs_attack</code>	When true, ASE fetches attack list from ABS and blocks access by clients in the attack list. When false, attack list is not downloaded.
<code>abs_attack_request_minute</code>	Time interval in minutes at which ASE fetches ABS attack list. The default value is 10 minutes.
Alerts and reports	
<code>enable_email</code>	When true, send email notifications. See Configure email notifications 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.
sender_email	Email address for sending email alerts and reports.
sender_password	Password of sender's email account.
receiver_email	Email address to notify about alerts and reports See email alerts for more information.
ASE server resource utilization	
cpu_usage	Percentage threshold value of CPU utilization. See email alerts for more information.
memory_usage	Percentage threshold value of memory usage. email alerts alerts for more information.
filesystem_size	Percentage threshold value of filesystem capacity. See email alerts for more information.
buffer_size	Customizable payload buffer size to reduce the number of iterations required for reading and writing payloads. Default value is 16KB. Minimum is 1KB and maximum is 32KB.

```

; 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 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_port=587
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

; Defines running mode for API Security Enforcer (Allowed values are inline
or sideband).
mode=inline

; keystore password
keystore_password=OBF:AES:sRNp0W7sSilzrReXeHodKQ:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerMPV

; WARNING! Following two are internal configurations. You should not change
these values.

; IPC UNIX domain socket between controller and balancer.
controller_balancer_unixsocket=/tmp/ase.sock

; IPC UNIX domain socket between balancer processes.
inter_balancer_unixsocket=/tmp/ase_ipc.sock

```

- [API naming guidelines](#)
- [Defining an API – API JSON configuration file](#)

Parent topic: [Sideband API Security Enforcer](#)

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

Parent topic: [ASE configuration - ase.conf](#)

Parent topic: [ASE configuration - ase.conf](#)

Defining an API – API JSON configuration file

The API JSON file parameters define the behavior and properties of your API. The sample API JSON files shipped with ASE can be changed to your environment settings and are populated with default values.

The following table describes the JSON file parameters:

Parameter	Description
-----------	-------------

protocol	API request type with supported values of: http - HTTP
url	The value of the URL for the managed API. You can configure up to three levels of sub-paths. For example, "/shopping"- name of a 1 level API "/shopping/electronics/phones" - 3 level API "/" - entire server (used for ABS API Discovery or load balancing)
hostname	Hostname for the API. The value cannot be empty. "*" matches any hostname.
Configure the client identifiers (for example, cookie, API key, OAuth2 token) used by the API	
cookie	Name of cookie used by the backend servers.
cookie_idle_timeout logout_api_enabled cookie_persistence_enabled	N/A
oauth2_access_token	When true, ASE captures OAuth2 Access Tokens. When false, ASE does not look for OAuth2 Tokens. Default value is false. For more information, see Configuring OAuth2 Token .
apikey_qs	When API key is sent in the query string, ASE uses the specified parameter name to capture the API key value. For more information, see Configuring API keys .
apikey_header	When API key is part of the header field, ASE uses the specified parameter name to capture the API key value. For more information, see Configuring API keys .
login_url	Public URL used by a client to connect to the application.

<code>enable_blocking</code>	When true, ASE blocks all types of attack on this API. When false, no attacks are blocked. Default value is false.
<code>api_mapping</code>	N/A
API pattern enforcement <code>protocol_allowed</code> <code>http_redirect</code> <code>methods_allowed</code> <code>content_type_allowed</code> <code>error_code</code> <code>error_type</code> <code>error_message_body</code>	N/A
Flow control <code>client_spike_threshold</code> <code>client_connection_queuing</code>	N/A
<code>api_memory_size</code>	Maximum ASE memory allocation for an API. The default value is 128 MB. The data unit can be MB or GB.
Health_check <code>health_check_interval</code> <code>health_retry_count</code> <code>health_url</code>	N/A
<code>server_ssl</code>	N/A
Servers: <code>host</code> <code>port</code>	The IP address or hostname and port number of each backend server running the API.
<code>server_spike_threshold</code> <code>server_connection_quota</code>	N/A
Decoy Config <code>decoy_enabled</code> <code>response_code</code> <code>response_def</code> <code>response_message</code> <code>decoy_subpaths</code>	<p>When decoy_enabled is set to true, decoy sub-paths function as decoy APIs .</p> <p>response_code is the status code (for example 200) that ASE returns when a decoy API path is accessed.</p> <p>response_def is the response definition (for example OK</p>

	<p>) that ASE returns when a decoy API path is accessed.</p> <p>response_message is the response message (for example OK) that ASE returns when a decoy API path is accessed.</p> <p>decoy_subpaths is the list of decoy API sub-paths (for example shop/admin, shop/root)</p> <p>See Configuring API deception for details</p>
--	---

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
```

Parent topic: [ASE configuration - ase.conf](#)

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](#)
- [Capture client identifiers](#)
- [Manage whitelist and blacklist](#)
- [ASE-generated error messages for blocked requests](#)
- [Per API blocking](#)

Parent topic: [Sideband API Security Enforcer](#)

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.

Parent topic: [Activate API cybersecurity](#)

Capture client identifiers

ASE identifies attackers for HTTP(s) protocol 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

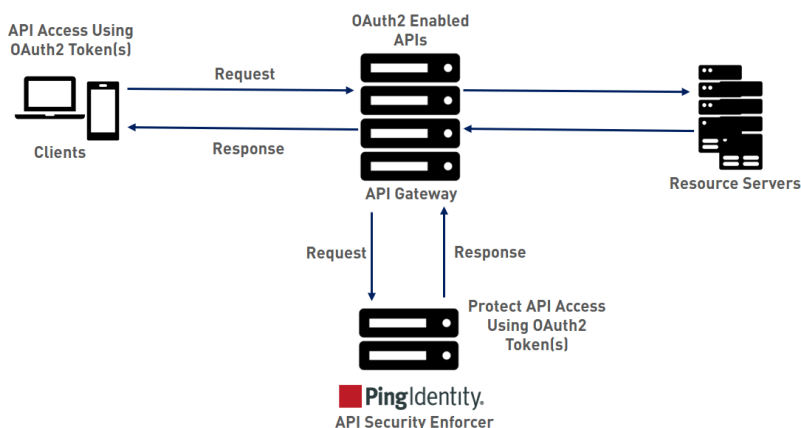
ASE supports capturing and blocking of OAuth2 tokens. To enable OAuth2 token capture, set the value of **oauth2_access_token** to true in the API JSON file. Here is a snippet of an API JSON file with OAuth2 token capture activated. To disable, change the value to false.

```
"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.

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.

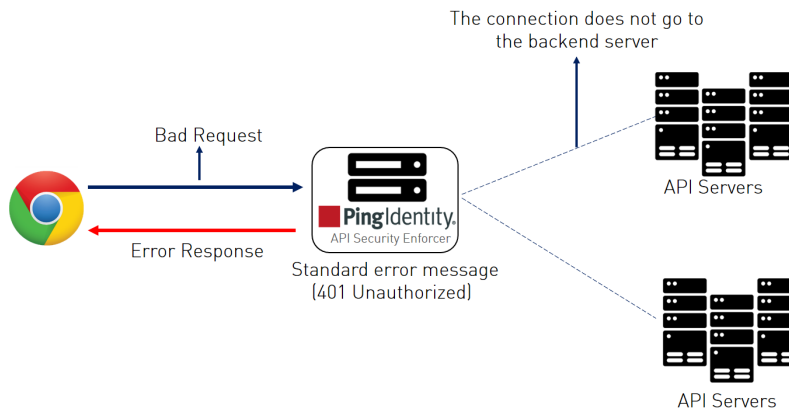
Parent topic: [Activate API cybersecurity](#)

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
Unknown API	503	Service Unavailable	Error: U
Unknown Hostname	503	Service Unavailable	Error: U
Malformed Request	400	Bad Request	Error: M
IP attack	401	Unauthorized	Error: U
Cookie attack	401	Unauthorized	Error: U
OAuth2 Token attack	401	Unauthorized	Error: U
API Key attack	401	Unauthorized	Error: U



Parent topic: [Activate API cybersecurity](#)

Parent topic: [Real-time API cybersecurity](#)

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.

Parent topic: [Activate API cybersecurity](#)

Parent topic: [Real-time API cybersecurity](#)


API deception environment

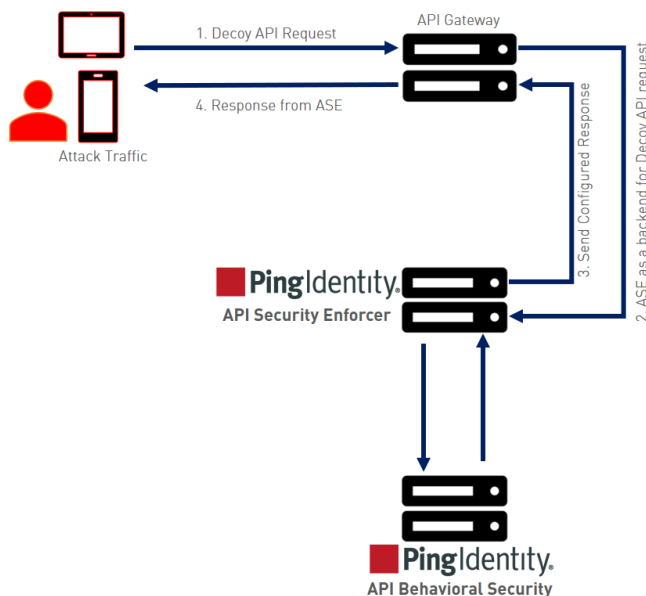
A decoy API is configured in ASE and the API gateway. It requires no changes to backend servers. It appears as part of the API ecosystem and is used to detect the attack patterns of hackers. When a hacker accesses a decoy API, ASE sends a predefined response (defined in the **response_message** parameter 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 14 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
```

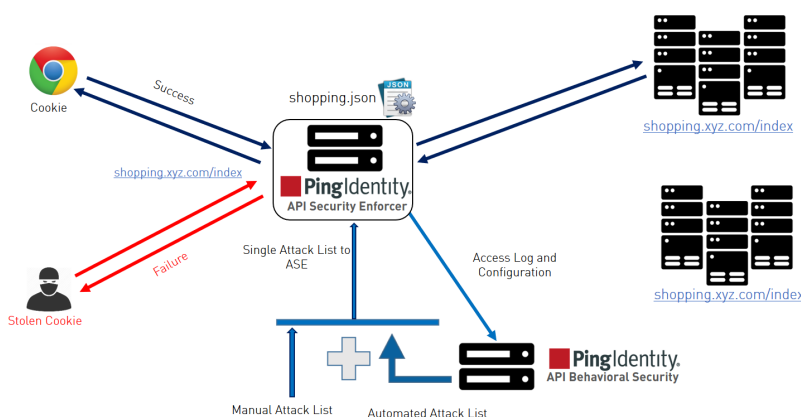
The blocked client is added to the blacklist which can be viewed by running the **view_blacklist** CLI command:

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin view_blacklist
Realtime Decoy Blacklist
1) type : ip, value : 100.100.1.1
```

Parent topic: [Sideband API Security Enforcer](#)

ASE configuration for ABS AI-based security

API Behavioral Security (ABS) 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 attack lists to ASE which blocks rogue clients on the list.



- [Configure ASE to ABS connectivity](#)
- [Manage ASE blocking of ABS detected attacks](#)

Parent topic: [Sideband API Security Enforcer](#)

Parent topic: [Inline API Security Enforcer](#)

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_abs` must be set to true in the `ase.conf` file. when ABS is in a different AWS security group, use a private IP address

The 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 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 all 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]}
```

Parent topic: [ASE configuration for ABS AI-based security](#)

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.

Parent topic: [ASE configuration for ABS AI-based security](#)

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 and API Key

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 and API key

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

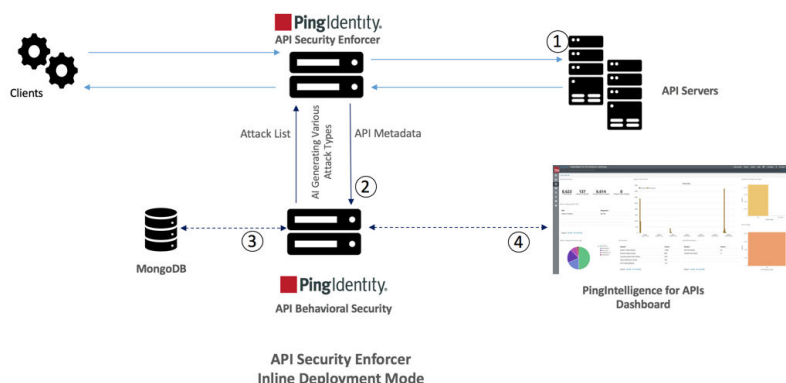
Syntax

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

Parent topic: [Sideband API Security Enforcer](#)

Inline API Security Enforcer

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 `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](#)
- [API routing](#)
- [Real-time API cybersecurity](#)
- [API deception environment](#)
- [ASE DoS and DDoS protection](#)
- [ASE configuration for ABS AI-based security](#)
- [CLI for inline ASE](#)

ASE configuration - ase.conf

ASE system level configuration entails modifying parameters in the `ase.conf` file located in the `config` directory. Some values have default settings which can be modified to support your application requirements. The parameter values and descriptions are included in the following table:

Parameter	Description
ASE mode	
<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> .
ASE ports	
<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.
<code>management_port</code>	Management port used for CLI and REST API management. The default value is 8010.
ASE administration and audit	
<code>admin_log_level</code>	The level of log detail captured. Options include: Fatal – 1, Error – 2, Warning – 3, Info – 4, Debug – 5
<code>enable_audit</code>	When set to true , ASE logs all actions performed in ASE in the audit log files. The default value is true .
<code>syslog_server</code>	Syslog server hostname or IPv4 address:port number . Leave this parameter blank if you do not want to generate for no syslog.
<code>hostname_refresh</code>	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.
<code>auth_method</code>	Authentication method used for administrator access. See Configuring Native and PAM Authentication for more information on the two options. <ul style="list-style-type: none"> • <code>ase::db</code> (Default - Native authentication) • <code>pam::ldap</code> (Linux-PAM Authentication with script)
<code>enable_ase_health</code>	When true, enables load balancers to perform a health check using the following URL: <code>http(s)://<ASE Name>/ase</code> where <code><ASE Name></code> is the ASE domain name. The default value is false.

	 Note: Do not configure the /ase URL in an API JSON file.
<code>enable_1G</code>	<p>When true, enable 1Gbps Ethernet support. The default value is true.</p>  Note: Only applicable when using a 1G NIC card
<code>http_ws_process</code>	<p>The number of HTTP or WebSocket processes. The default value is 1 and the maximum value is 6.</p>  Note: When running ASE in a cluster deployment, all nodes must have the same number of processes.
<code>https_wss_process</code>	<p>The number of HTTPS or secure WebSocket processes. The default value is 1 and the maximum value is 6.</p>  Note: When running ASE in a cluster deployment, all nodes must have the same number of processes.
<code>enable_access_log</code>	<p>When true, log client traffic request and response information. Default is true.</p>
<code>flush_log_immediate</code>	<p>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.</p>
<code>attack_list_memory</code>	<p>The amount of memory used for maintaining black and whitelists. The default value is 128 MB.</p>
<code>keystore_password</code>	<p>Password for the keystore. For more information on updating the keystore password, see Updating Keystore Password.</p>
ASE cluster	
<code>enable_cluster</code>	 When true, run the setup in cluster mode. The default value is false, run the setup in standalone mode.
Security	
<code>enable_sslv3</code>	<p>When true, enable SSLv3. Default value is false.</p>
<code>server_ca_cert_path</code>	<p>Location of the trusted CA certificates for SSL/TLS connections from clients to backend servers.</p> <p>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.</p> <p>For RHEL 7 CA certificates, the default path is: <code>/etc/pki/tls/certs</code>. Multiple certificates can be placed in this directory.</p>

enable_xff	When true, pass XFF header with originating IP address to the backend server.
enable_firewall	When true, activate the following API security features: <ul style="list-style-type: none"> • API mapping • API pattern enforcement • Connection drop using attack types • Flow control Default value is true
Real-time API security	
enable_ase_detected_attack	When true, activates the real-time security in ASE. ASE detects and blocks pattern enforcement violations, wrong API keys and clients probing decoy API and later accessing real APIs. The default value is false.
API deception	
decoy_alert_interval	The time interval between decoy API email alerts. The default value is 180 minutes. Maximum value is 1440 minutes (i.e. 24 hours).
AI-based API 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 the clients that are in the attack list. When false, attack list is not downloaded.
abs_attack_request_minute	Time interval in minutes at which ASE fetches ABS attack list. The default value is 10-minutes.
Alerts and reports	
enable_email	When true, send email notifications. See Configure email notifications for more information. The default value is false.
email_report	Time interval in days at which ASE sends reports. Minimum value is 1 and the maximum is 7-days. The default value is 1-day.
smtp_host	Hostname of SMTP server.
smtp_port	Port number of SMTP server.
sender_email	Email address for sending email alerts and reports.
sender_password	Password of sender's email account.
receiver_email	Email address to notify about alerts and reports

	See email alerts for more information.
ASE server resource utilization	
cpu_usage	Percentage threshold value of CPU utilization. See email alerts for more information.
memory_usage	Percentage threshold value of memory usage. See email alerts for more information.
filesystem_size	Percentage threshold value of filesystem capacity. See email alerts for more information.
buffer_size	Customizable payload buffer size to reduce the number of iterations required for reading and writing payloads. Default value is 16KB. Minimum is 1KB and maximum is 32KB.

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

```
; This is API Security Enforcer's main configuration file. This file is in
the standard .ini format.
; It contains ports, firewall, log, ABS flags. The comments start with a
semicolon (;).

; Defines 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_port=587
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

```

```

; Defines running mode for API Security Enforcer (Allowed values are inline
or sideband).
mode=inline

; keystore password
keystore_password=OBF:AES:sRNp0W7sSilzrReXeHodKQ:lXcvbBhKZgDTrjQOfOkzR2mpca4bTUcwPAuerMPv

; WARNING! Following two are internal configurations. You should not change
these values.

; IPC UNIX domain socket between controller and balancer.
controller_balancer_unixsocket=/tmp/ase.sock

; IPC UNIX domain socket between balancer processes.
inter_balancer_unixsocket=/tmp/ase_ipc.sock

```

- [API naming guidelines](#)
- [Define an API – API JSON configuration file](#)

Parent topic: [Inline API Security Enforcer](#)

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

Parent topic: [ASE configuration - ase.conf](#)

Parent topic: [ASE configuration - ase.conf](#)

Define 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: ws - WebSocket ; http - HTTP
url	The value of the URL for the managed API. You can configure up paths. For example, "/shopping"- name of a 1 level API "/shopping/electronics/phones" – 3 level API "/" – entire server (used for ABS API Discovery or load balancing)
hostname	Hostname for the API. The value cannot be empty.

	"*" matches any hostname.
<code>cookie</code>	Name of cookie used by the backend servers.
<code>cookie_idle_timeout</code>	The amount of time a cookie is valid – for example 20m for 20 minutes. The time duration formats include: s: seconds, m: minutes, h: hour, d: day <ul style="list-style-type: none"> • w: week • mnt: month • yr: year
<code>logout_api_enabled</code>	When true, ASE expires cookies when a logout request is sent.
<code>cookie_persistence_enabled</code>	When true, the subsequent request from a client is sent to the server that responded.
<code>oauth2_access_token</code>	When true, ASE captures OAuth2 Access Tokens. When false, ASE does not look for OAuth2 Tokens. Default value is <code>true</code> . For more information, see Configuring OAuth2 Token .
<code>apikey_qs</code>	When API Key is sent in the query string, ASE uses the specified field to capture the API key value. For more information, see Configuring API Keys .
<code>apikey_header</code>	When API Key is part of the header field, ASE uses the specified field to capture the API key value. For more information, see Configuring API Keys .
<code>login_url</code>	Public URL used by a client to connect to the application.
<code>enable_blocking</code>	When true, ASE blocks all types of attack on this API. When false, ASE does not block attacks. Default value is <code>false</code> .
<code>api_memory_size</code>	Maximum ASE memory allocation for an API. The default value is 128 MB. The data unit can be MB or GB.
<code>health_check</code>	When true, enable health checking of backend servers. When false, no health checks are performed. Ping Identity recommends setting this parameter as true.
<code>health_check_interval</code>	The interval in seconds at which ASE sends a health check to the backend server status.
<code>health_retry_count</code>	The number of times ASE queries the backend server status after a failed response.
<code>health_url</code>	The URL used by ASE to check backend server status.

server_ssl	When set to true , ASE connects to the backend API server over SSL. ASE uses TCP to connect to the backend server.
Servers: host port server_spike_threshold server_connection_quota	The IP address or hostname and port number of each backend server. See REST API Protection from DoS and DDoS for information on these parameters.
API Mapping: internal_url	Internal URL is mapped to the public external URL. See API Name Mapping – Protect Internal URLs for more information.
The following API Pattern Enforcement parameters only apply when API Firewall is activated	
Flow Control client_spike_threshold server_connection_queueing bytes_in_threshold bytes_out_threshold	ASE flow control ensures that backend API servers are protected from (DDoS, traffic spike) in API traffic. See WebSocket API Protection from DoS and DDoS for information on these parameters.
protocol_allowed	List of accepted protocols Values can be HTTP, HTTPS, WS, WSS.  Note: When Firewall is enabled, protocol_allowed overrides the protocol parameter.
http_redirect response_code response_def https_url	Redirect unencrypted HTTP requests to https_url , the FQDN of the secure connection. See Configuring Pattern Enforcement for details.
methods_allowed	List of accepted REST API methods. Possible values are: GET, POST, PUT, DELETE, HEAD
content_type_allowed	List of content types allowed. Multiple values cannot be listed. For example, application/json.
error_code error_type error_message_body	Error message generated by ASE after blocking a client See ASE Detected Error Messages for details
Decoy Config decoy_enabled response_code response_def response_message decoy_subpaths	When decoy_enabled is set to true, decoy sub-paths function as follows: response_code is the status code (for example, 200) that ASE returns when a decoy API path is accessed. response_def is the response definition (for example OK) that ASE returns when a decoy API path is accessed. response_message is the response message (for example OK) that ASE returns when a decoy API path is accessed.

	decoy_subpaths is the list of decoy API sub-paths (for example root) See Configuring API deception 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.

A sample.json file for a WebSocket API:

```

{
  "api_metadata": {
    "protocol": "ws",
    "url": "/app",
    "hostname": "*",
    "cookie": "JSESSIONID",
    "cookie_idle_timeout": "200m",
    "logout_api_enabled": false,
    "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": ""
    },
  },
  "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": []
}
}
}

```

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
```

Parent topic: [ASE configuration - ase.conf](#)

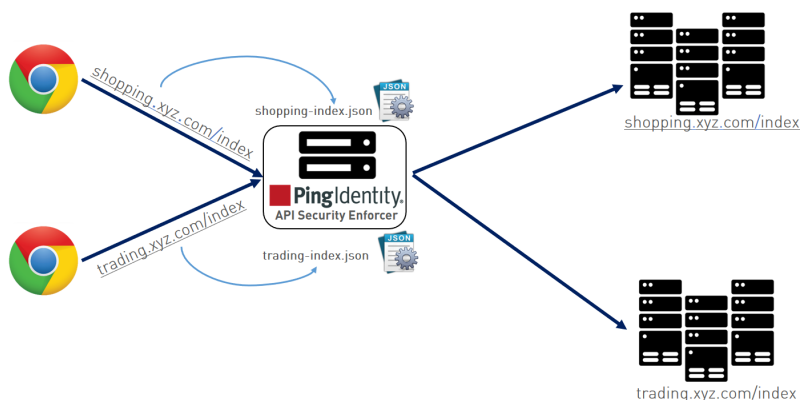
API routing

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:

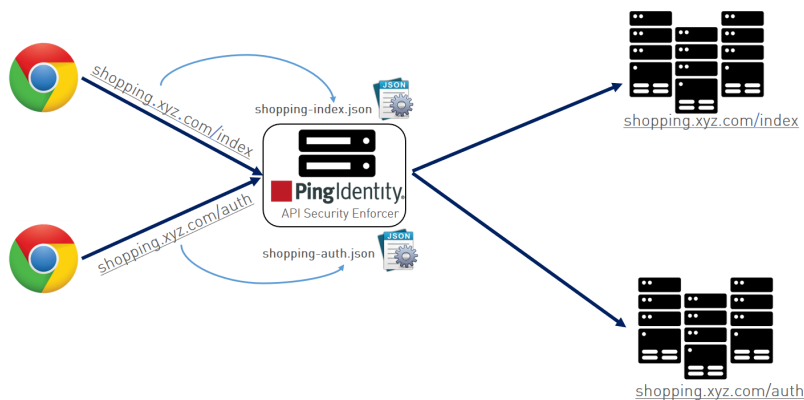
```
"api_metadata": {
  "protocol": "https",
  "url": "/index",
  "hostname": "shopping.xyz.com",
  "cookie": "JSESSIONID",
  "cookie_idle_timeout": "200m",
  "logout_api_enabled": true,
  "cookie_persistence_enabled": false,
```


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`.

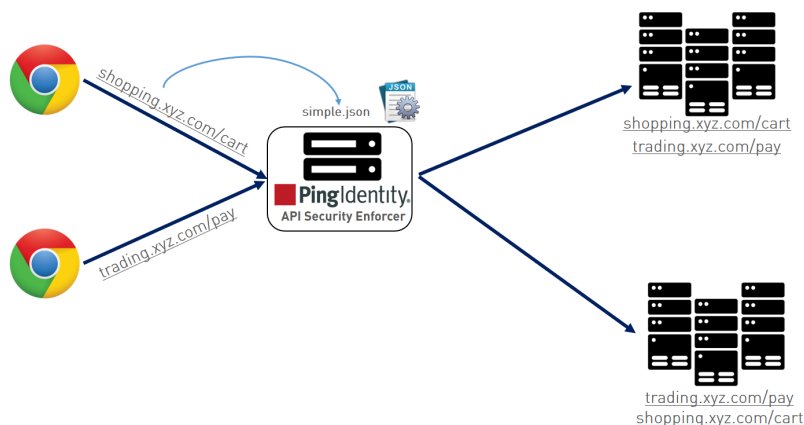


Simple routing

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.

Parent topic: [Inline API Security Enforcer](#)

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 name mapping – hide internal URLs](#)
- [Capturing client identifiers](#)
- [Manage whitelist and blacklist](#)
- [Map server error messages to custom error messages](#)
- [ASE-generated error messages for blocked requests](#)
- [Per API blocking](#)

Parent topic: [Inline API Security Enforcer](#)

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](#))

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 **protocol** parameter in the beginning of the API JSON file

Detection of attacks for pattern enforcement violation

The following is a snippet of access log file showing what is logged when a connection is blocked based on any pattern enforcement violation.



Note: Make sure that ASE detected attacks are enabled.

The following example shows a method violation for an OAuth2 token:

```
[Fri Aug 10 15:59:12:435 2018] [thread:14164] [info]
[connectionid:1681692777] [seq:1] [connectinfo:100.100.1.5:36839]
```

```
[type:request] [api_id:shop] PATCH /shopapi/categories/list HTTP/1.1
User-Agent: curl/7.35.0
Accept: */*
Host: app
Content-Type: application/text
Cookie: JSESSIONID=ebcookie
Authorization: Bearer OAuthTokenusemethoid12345
[Fri Aug 10 15:59:12:435 2018] [thread:14164] [info]
[connectionid:1681692777] [seq:1] [connectinfo:100.100.1.5:36839]
[type:connection_drop] [enforcement:method] [api_id:shop] PATCH /shopapi/
categories/list HTTP/1.1
User-Agent: curl/7.35.0
Accept: */*
Host: app
Content-Type: application/text
Cookie: JSESSIONID=ebcookie
Authorization: Bearer OAuthTokenusemethoid12345
```

Violations logged in the ASE access log files are sent to API Behavioral Security engine for further analysis and reporting.

Parent topic: [Real-time API cybersecurity](#)

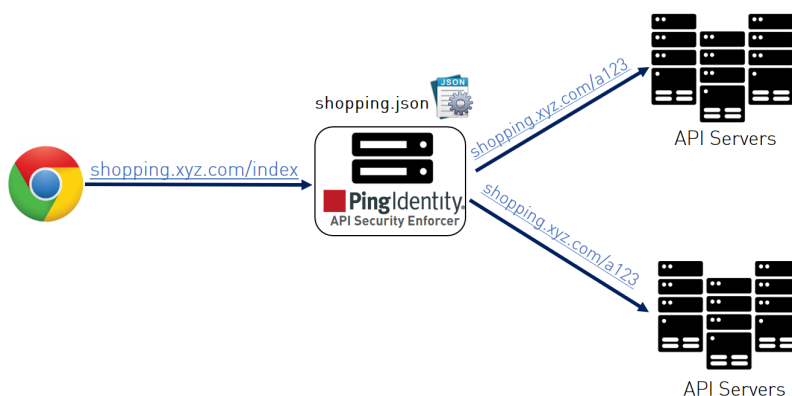
API name mapping – hide internal URLs

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:



Parent topic: [Real-time API cybersecurity](#)

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.

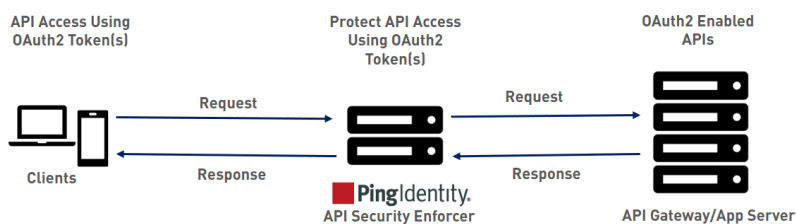
Configure ASE support for OAuth2 tokens

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.

Parent topic: [Real-time API cybersecurity](#)

Manage whitelist and blacklist

ASE maintains two types of lists:

- **Whitelist** – List of “safe” IP addresses, cookies, OAuth2 Tokens, or API keys that are not 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 attacks (for example data exfiltration)
 - ASE detected attacks (for example invalid method, decoy API accessed)
 - List of “bad” clients manually generated by CLI

Manage whitelists

Valid operations for OAuth2 Tokens, cookies, IP addresses and API keys on a whitelist 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 : token1.4
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
/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

```
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist ip 1.1.1.1
ip 1.1.1.1 added to blacklist
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist cookie
JSESSIONID ad233edqsd1d23redwefew
cookie JSESSIONID ad233edqsd1d23redwefew added to blacklist
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist token
ad233edqsd1d23redwefew
token ad233edqsd1d23redwefew added to blacklist
/opt/pingidentity/ase/bin/cli.sh -u admin -p admin add_blacklist api_key
AccessKey b31dfa4678b24aa5a2daa06aba1857d4
api_key AccessKey b31dfa4678b24aa5a2daa06aba1857d4 added to blacklist
```

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 : 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 : token1.4
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 : token1.4
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.

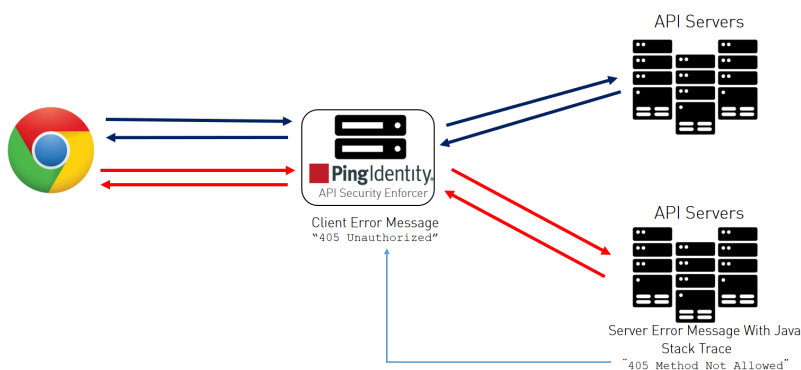
Blacklist to whitelist transition

When you delete a blacklist entry which was detected by ABS or ASE, it is automatically added to the whitelist and no longer blocked by ASE. However, CLI added entries deleted from the blacklist are not added to the whitelist. When the blacklist is cleared, list entries are not transitioned to the whitelist.

Parent topic: [Real-time API cybersecurity](#)

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.

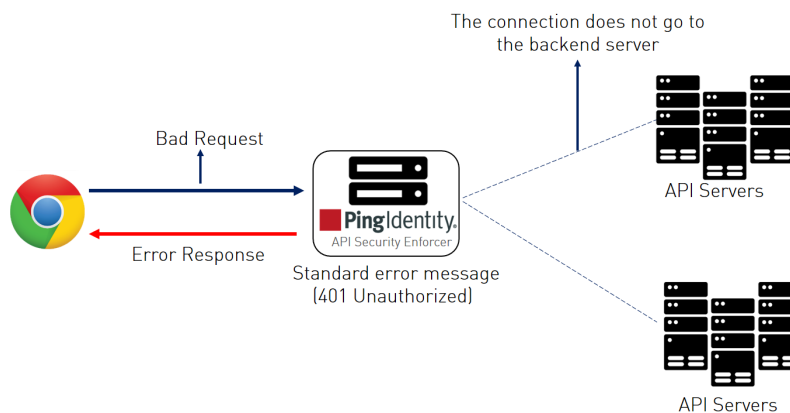
Parent topic: [Real-time API cybersecurity](#)

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
Unknown API	503	Service Unavailable	Error: U
Unknown Hostname	503	Service Unavailable	Error: U
Malformed Request	400	Bad Request	Error: M
IP attack	401	Unauthorized	Error: U
Cookie attack	401	Unauthorized	Error: U
OAuth2 Token attack	401	Unauthorized	Error: U
API Key attack	401	Unauthorized	Error: U



Parent topic: [Activate API cybersecurity](#)

Parent topic: [Real-time API cybersecurity](#)

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.

Parent topic:[Activate API cybersecurity](#)

Parent topic:[Real-time API cybersecurity](#)

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 **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 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.

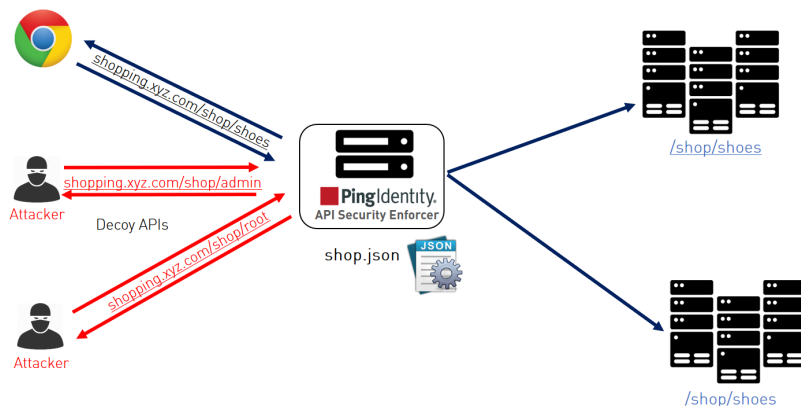
ASE Provides the following decoy API types:

- In-context decoy APIs
- Out-of-context decoy APIs
- [In-context decoy API](#)
- [Out-of-context decoy API](#)
- [Real-time API deception attack blocking](#)

Parent topic:[Inline API Security Enforcer](#)

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.

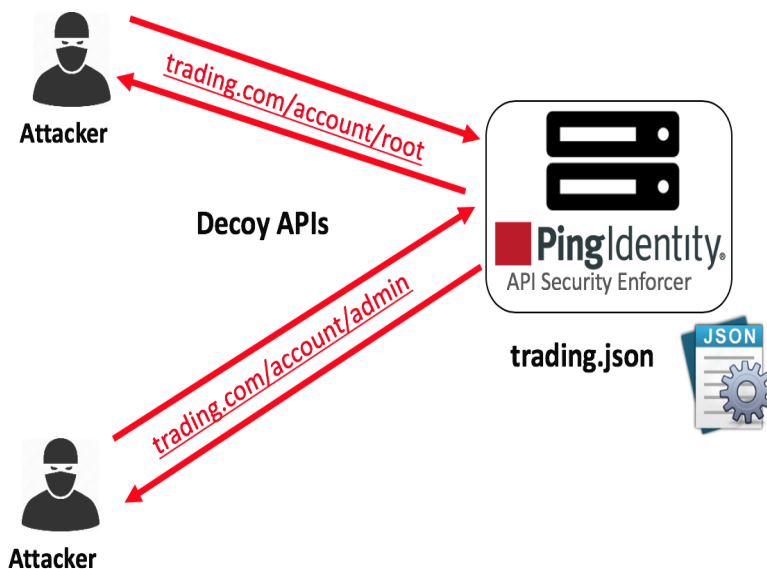
An in-context decoy API can have a maximum of 32 sub-paths configured for an API.

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.

Parent topic: [API deception environment](#)

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_subpaths": [
        ]
    }
}

```

Decoy API Configuration

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

```

Parent topic: [API deception environment](#)

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

```

Parent topic: [API deception environment](#)

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 WebSoc
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](#)
- [WebSocket API protection from DoS and DDoS](#)
- [Server connection queuing for REST and WebSocket APIs](#)

Parent topic:[Inline API Security Enforcer](#)

REST API protection from DoS and DDoS

flow control protects REST API servers using four control variables which are independently configured. By default, no flow control is enabled.

Variable	Description
Configured once in every API JSON file	
client_spike_threshold	Maximum requests per time-period from a single client IP to a REST API. Time can be in seconds, minutes or hours.
server_connection_queueing	When true, queue API connection requests when all backend reach server connection quota. Default value is false.
Configured for each server in every API JSON file	
server_connection_quota	Maximum number of concurrent connections to a specific 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 server. Time can be in seconds, minutes, or hours.

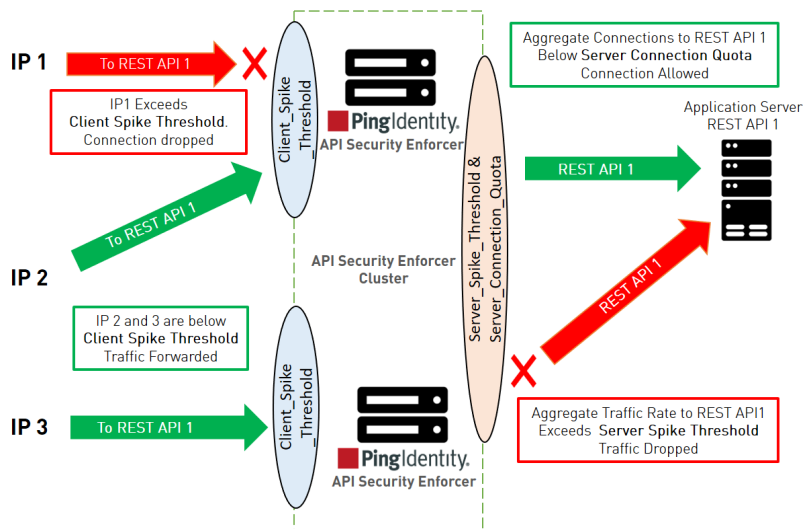
Client flow control monitors incoming traffic from each client connection and drops the session when traffic limits are exceeded. The diagram shows the following client scenarios:

- 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 on a specific server.
<code>server_connection_quota</code>	Maximum number of concurrent sessions from ASE cluster node API on a specific server.



Note: You can also configure server connection quota and server spike threshold separately for each backend server.

JSON configuration for REST API flow control

ASE flow control is configured separately for each API using the API JSON file. Here are the flow control related definitions in an API JSON file:

```
{
  "api_metadata": {
    "protocol": "http",

    "flow_control": {
      "client_spike_threshold": "0/second",
      "server_connection_queueing" : false
    },
    "servers": [
      {
        "host": "127.0.0.1",
        "port": 8080,
        "server_spike_threshold": "100/second",
        "server_connection_quota": 20
      },
      {
        "host": "127.0.0.1",
        "port": 8081,
        "server_spike_threshold": "200/second",
        "server_connection_quota": 40
      }
    ]
  }
}
```

The flow control section includes definitions which apply globally across the API definition and include `client_spike_threshold` and `server_connection_queueing`. Server specific definitions include

server_spike_threshold and **server_connection_quota** which are configured on each individual server. The default is no flow control with all values set to zero. Note that different values can be specified for each server for **server_connection_quota** and **server_spike_threshold**.



Note: If server connection quota is set to zero for one server, then it must be zero for all other servers in the API JSON definition.

Flow control CLI for REST API

ASE CLI can be used to update flow control parameters:

Update client spike threshold:

Enter the following command to update the client spike threshold:

```
update_client_spike_threshold {api_id} {+ve digit/(second|minute|hour)}
```

For example: `update_client_spike_threshold shop_api 5000/second`

Update server spike threshold

Enter the following command to update the server spike threshold:

```
update_server_spike_threshold {api_id} {host:port} {+ve digit/(second|minute|hour)}
```

For example: `update_server_spike_threshold shop_api 5000/second`

Update server connection quota

```
update_server_connection_quota {api_id} {host:port}{+ve digit}
```

For example: `update_server_connection_quota shop_api 5000`



Note: API security must be enabled for ASE flow control to work. For more information on enabling API security, see [Enable API security](#)

Parent topic: [ASE DoS and DDoS protection](#)

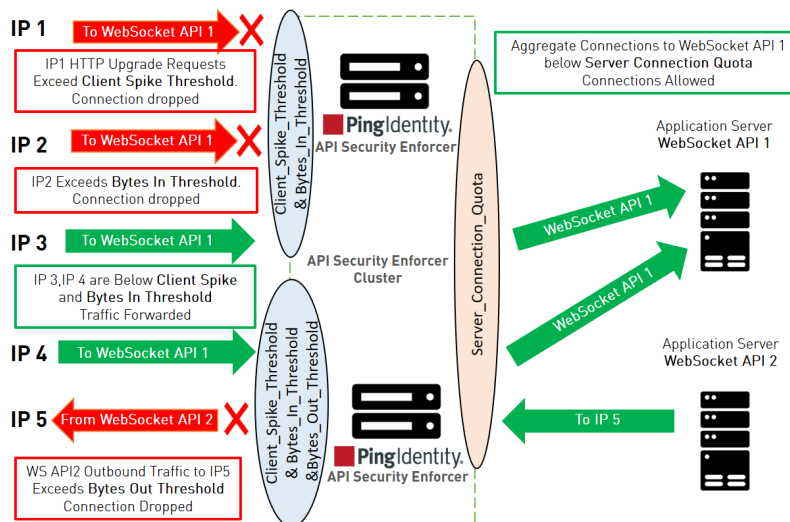
WebSocket API protection from DoS and DDoS

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

Variable	Description
Configured once in every API JSON file	
client_spike_threshold	Maximum number of HTTP requests per time-period from a single IP to a specific WebSocket API.

	Time can be in seconds, minutes or hours.
bytes_in_threshold	Maximum number of bytes per time-period from a single IP to an ASE node. Time can be in seconds, minutes or hours.
bytes_out_threshold	Maximum number of bytes per time-period sent from an ASE node to a single IP. Time can be in seconds, minutes or hours.
server_connection_queueing	When true, queue connection requests when all backend servers reach the server connection quota. The default value is false.
Configured for each server in every API JSON file	
server_connection_quota	Maximum number of concurrent connections to a specific WebSocket API on a server. Prevents aggregate connections from one or a cluster of ASE nodes from overloading a WebSocket API on a specific server.

The following diagram shows the effect of the parameters on traffic flow through ASE. In the diagram, client-side flow control is managed by **client_spike_threshold**, **bytes_in_threshold**, and **bytes_out_threshold**. The **bytes_out** threshold protects against data exfiltration. Server flow control is regulated by **server_connection_quota**.



Client flow control monitors incoming traffic from each client connection and drops sessions when HTTP request or bytes in threshold limits are exceeded. In addition, outbound traffic from each ASE Node is monitored to protect against exfiltration. The diagram shows client scenarios including:

- IP1 sending HTTP request volumes which exceed the **client_spike_threshold** value. ASE 1 sends an error message and terminates the session to stop the attack.

- IP2 sending WebSocket streaming traffic volumes which exceed the **bytes_in_threshold** limits. ASE 1 sends an error message and terminates the session to stop the traffic.
- IP3 and IP4 within client spike threshold and bytes in threshold criteria and requests are forwarded to the backend server.
- Traffic from ASE 2 to IP5 exceeds the bytes out threshold value. ASE blocks the traffic and drops the client session.

The server-side flow control provides the ability to control session count to an API on an application server. **server_connection_quota** sets the maximum number of concurrent connections that can be established to an API on a server. The concurrent connections are the aggregate connections from all ASE nodes forwarding traffic to the specified API on a given server.

Example:

Here is an example with a hypothetical deployment for the Application Server in the previous diagram.

Variable	Configured value
client_spike_threshold	50,000 requests per second per IP
bytes_in_threshold	2000 bytes per second per IP
bytes_out_threshold	1000 bytes per second per server
server_connection_quota	20,000 concurrent connections per server
server_connection_queueing	true

Client flow control permits a maximum of 50,000 HTTP requests/second from an individual IP. If IP 1, 2, or 3 exceeds the 50,000/second limit, ASE drops the client session. Otherwise, all requests are passed to the backend servers.

Client flow control allows a maximum of 2,000 bytes/second from each WebSocket client connection to an ASE node. If IP 1, 2, or 3 exceeds the 2,000 bytes/second limit, ASE drops the client session. Otherwise, all requests are passed to the backend servers.

Server flow control allows 20,000 concurrent connections to WebSocket API 1 on the application server. If the sum of connections from the ASE cluster nodes (i.e. ASE 1 + ASE 2 connection count) to WebSocket API1 exceeds 20,000, then ASE will queue the request for a time-period since **server_connection_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 API – API JSON configuration file](#) 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](#).

Parent topic: [ASE DoS and DDoS protection](#)

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 $i=n$

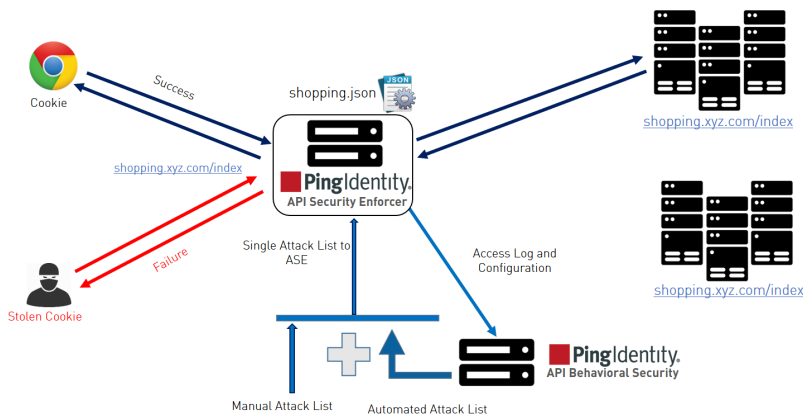
For example, if two backend servers have connection quota set as 20,000 and 40,000 connections, then the connections are served in a ratio of $20000 / (20000+40000)$ and $40000 / (20000+40000)$, that is, in the ratio of 1/3 and 2/3 for the respective servers. The weighted ratio of connection distribution 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.

Parent topic: [ASE DoS and DDoS protection](#)

ASE configuration for ABS AI-based security

API Behavioral Security (ABS) 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 attack lists to ASE which blocks rogue clients on the list.



- [Configure ASE to ABS connectivity](#)
- [Manage ASE blocking of ABS detected attacks](#)

Parent topic:[Sideband API Security Enforcer](#)

Parent topic:[Inline API Security Enforcer](#)

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_abs` must be set to true in the `ase.conf` file. when ABS is in a different AWS security group, use a private IP address

The 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 the ABS nodes. The same value has to be configured in ABS MongoDB database. This value is obfuscated during the start of ASE.
<code>secret_key</code>	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.
<code>enable_ssl</code>	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.
<code>abs_ca_cert_path</code>	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]}
```

Parent topic: [ASE configuration for ABS AI-based security](#)

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.

Parent topic: [ASE configuration for ABS AI-based security](#)

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 and API Key

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 and API key

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
```

Parent topic: [Inline API Security Enforcer](#)

REST API for inline and sideband ASE

ASE REST API allows you to programmatically 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 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	<Access Key>
x-ase-secret-key	<Secret Key>
Accept	application/json

Create API (POST)

Request

POST	/v3/ase/api?api_id=sample_api
Content-Type	application/json
x-ase-access-key	<Access Key>
x-ase-secret-key	<Secret Key>
Accept	application/json

REST API request

```
{
  "api_metadata": {
    "protocol": "http",
    "url": "/your_rest_api",
    "hostname": "*",
    "cookie": "",
    "cookie_idle_timeout": "200m",
  }
}
```

```

"logout_api_enabled": false,
"cookie_persistence_enabled": false,
"oauth2_access_token": false,
"apikey_qs": "",
"apikey_header": "",
"login_url": "",
"enable_blocking": true,
"api_mapping": {
"internal_url": ""
},
"api_pattern_enforcement": {
"protocol_allowed": "",
"http_redirect": {
"response_code": "",
"response_def": "",
"https_url": ""
},
"methods_allowed": [],
"content_type_allowed": "",
"error_code": "401",
"error_def": "Unauthorized",
"error_message_body": "401 Unauthorized"
},
"flow_control": {
"client_spike_threshold": "0/second",
"server_connection_queueing": false
},
"api_memory_size": "128mb",
"health_check": true,
"health_check_interval": 60,
"health_retry_count": 4,
"health_url": "/health",
"server_ssl": false,
"servers": [
{
"host": "127.0.0.1",
"port": 8080,
"server_spike_threshold": "0/second",
"server_connection_quota": 0
},
{
"host": "127.0.0.1",
"port": 8081,
"server_spike_threshold": "0/second",
"server_connection_quota": 0
}
],
"decoy_config": {
"decoy_enabled": false,
"response_code": 200,
"response_def": "",
"response_message": "",
"decoy_subpaths": []
}
}

```

```
}
}
```

WebSocket API request

```
{
  "api_metadata": {
    "protocol": "ws",
    "url": "/your_websocket_api",
    "hostname": "*",
    "cookie": "",
    "cookie_idle_timeout": "200m",
    "logout_api_enabled": false,
    "cookie_persistence_enabled": false,
    "oauth2_access_token": false,
    "apikey_qs": "",
    "apikey_header": "",
    "login_url": "",
    "enable_blocking": true,
    "api_mapping": {
      "internal_url": ""
    },
    "api_pattern_enforcement": {
      "protocol_allowed": "",
      "http_redirect": {
        "response_code": "",
        "response_def": "",
        "https_url": ""
      },
      "methods_allowed": [],
      "content_type_allowed": "",
      "error_code": "401",
      "error_def": "Unauthorized",
      "error_message_body": "401 Unauthorized"
    },
    "flow_control": {
      "client_spike_threshold": "0/second",
      "bytes_in_threshold": "0/second",
      "bytes_out_threshold": "0/second",
      "server_connection_queueing": false
    },
    "api_memory_size": "128mb",
    "health_check": true,
    "health_check_interval": 60,
    "health_retry_count": 4,
    "health_url": "/health",
    "server_ssl": false,
    "servers": [
      {
        "host": "127.0.0.1",
        "port": 8080,
        "server_connection_quota": 0
      },

```



```

{
  "host": "127.0.0.1",
  "port": 8081,
  "server_connection_quota": 0
}
],
"decoy_config": {
  "decoy_enabled": false,
  "response_code": 200,
  "response_def": "",
  "response_message": "",
  "decoy_subpaths": []
}
}
}

```

Response

HTTP Code	Status	Content body (application/json)
200	success	{ "status" : "success" , "status_message" : "success" }
403	fail	{ "status" : "api_already_exists" , "status_message" : "api sample_api already exists" }
403	fail	{ "status" : "validation_error" , "status_message" : "<etailed validation error description" }

Read API (GET)

Request

GET	/v3/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	<p>REST API</p> <pre> { "api_metadata": { "protocol": "http", "url": "/your_rest_api", "hostname": "*", "cookie": "", "cookie_idle_timeout": "200m", "logout_api_enabled": false, "cookie_persistence_enabled": false, "oauth2_access_token": false, "apikey_qs": "", "apikey_header": "", "login_url": "", "enable_blocking": true, "api_mapping": { "internal_url": "" }, "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": { </pre>

HTTP Code	Status	Content body (application/json)
		<pre> "decoy_enabled": false, "response_code": 200, "response_def": "", "response_message": "", "decoy_subpaths": [] } } } WebSocket API { "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, </pre>

HTTP Code	Status	Content body (application/json)
		<pre> "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": [] } } } </pre>
404	not found	<pre> {"status": "api_not_found", "status_message": "api sample_api does not exist"} </pre>

List API (GET)

Request

GET	/v3/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>

HTTP Code	Status	Content body (application/json)
404	not found	<pre>{"status" : "api_not_found" , "status_message" : "api sample_api does not exist"}</pre>

Update API (PUT)

Request

PUT	/v3/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	/v3/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	/v3/ase/server?api_id=<api>
Content-Type	application/json
x-ase-access-key	<Access Key>
x-ase-secret-key	<Secret Key>
Accept	application/json

REST API request

```
{
  "server":
  {
    "host": "192.168.1.100",
    "port": 8080,
    "server_spike_threshold": "1/second",
    "server_connection_quota": 100
  }
}
```

WebSocket API Request

```
{
  "server":
  {
    "host": "192.168.1.100",
    "port": 8080,
    "server_connection_quota": 100
  }
}
```

Response

HTTP Code	Status	Content body (application/json)
200	success	{ "status" : "success" , "status_message" : "success" }
404	fail	{ "status" : "api_not_found" , "status_message" : "api sample_api does not exist" }
403	fail	{ "status" : "validation_error" , "status_message" : "detailed info about validation error" }
403	fail	{ "status" : "server_exists" , "status_message" : "server already exists" }

Read server (GET)

Request

GET	/v3/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>REST API</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>WebSocket API</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	/v3/ase/server?api_id=<api>
Content-Type	application/json
x-ase-access-key	<Access Key>
x-ase-secret-key	<Secret Key>
Accept	application/json

```
{
  "server":
  {
    "host" : "192.168.1.100",
    "port" : 8080
  }
}
```

Response

HTTP Code	Status	Content body (application/json)
200	success	{ "status" : "success" , "status_message" : "success" }
404	fail	{ "status" : "api_not_found" , "status_message" : "api sample_api does not exist" }
404	fail	{ "status" : "server_not_found" , "status_message" : "server does not exist" }
403	fail	{ "status" : "validation_error" , "status_message" : "detailed info about json validation error" }

Read cluster (GET)**Request**

GET	/v3/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	/v3/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	/v3/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	<pre>{ "status" : "enabled/ disabled", "status_message" : "Ok" }</pre>

Update firewall status (POST)**Request**

POST	/v3/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>If there is a status change</p> <pre>{ "status" : "enabled/disabled", "status_message" : "Firewall is now enabled/ disabled" }</pre> <p>If there is no change in status</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	/v3/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 ljkhasioutfdqbjfsdmakhflia 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	/v3/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" : "ljkhasioutfdqbjfsfdmakhflia"
}

```

Response

Status code	Response body
200 OK	Cookie JSESSIONID ljkhasioutfdqbjfsfdmakhflia deleted from blacklist
403 Forbidden	Cookie JSESSIONID ljkhasioutfdqbjfsfdmakhflia 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	/v3/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	/v3/ase/firewall/blacklist?tag=
Tags	tag=all (default is all) <ul style="list-style-type: none"> • all • manual • abs_generated • invalid_content_type • invalid_method • invalid_protocol • decoy
x-ase-access-key	<Access Key>
x-ase-secret-key	<Secret Key>
Accept	application/json

Response

Status code	Response body
200 OK	<pre>{ "manual_blacklist" : [{ "type" : "cookie", "name" : "JSESSIONID", "value" : "ljkhasiosalia", }, { "type" : "ip", "value" : "1.1.1.1", }], "abs_generated_blacklist" : [{ "type" : "cookie", "name" : "JSESSIONID", "value" : "ljkhasisadosalia", }, { "type" : "ip", "value" : "1.1.1.2", }] }</pre>
403 Forbidden	Cookie JSESSIONID ljkhasioutfdqbjfsfdmakhflia 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	/v3/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" : "ljkhasioutfdqbjfsdmakhflia"
}

```

Response

Status code	Response body
200 OK	Cookie JSESSIONID ljkhasioutfdqbjfsdmakhflia added to whitelist
403 Forbidden	Cookie JSESSIONID ljkhasioutfdqbjfsdmakhflia 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	/v3/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" : "ljkhasioutfdqbjfsdmakhflia"
}

```

Response

Status code	Response body
200 OK	Cookie JSESSIONID ljkhasioutfdqbjfsdmakhflia added to whitelist
403 Forbidden	Cookie JSESSIONID ljkhasioutfdqbjfsdmakhflia 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	/v3/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	/v3/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	{ "whitelist" : [{

Status code	Response body
	<pre>"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	/v3/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>Flow control for REST API</p> <pre>{ "api_id": "api_name" "flow_control": { "client_spike_threshold": "0/second", "server_connection_queueing": false } }</pre> <p>Flow control for WebSocket API</p>

HTTP code	Status	Content body (application/json)
		<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" : "<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	/v3/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>Flow control for REST APIs</p> <pre>{ "api_id": "api_name" "flow_control": { "client_spike_threshold": "0/second", "server_connection_queueing": false } }</pre> <p>Flow control for WebSocket APIs</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" : "<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	/v3/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" : "<detailed validation error description" }</pre> }
404	fail	{ <pre>"status" : "api_not_found" , "status_messag e" : "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	Resource
------	---------	--------	----------

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:

```
2017-01-13 10:45:12 | cli | delete_api | username=admin, api_id=cart
2017-01-13 10:46:13 | rest_api | GET /v3/ase/cluster | x-ase-access-key=admin, x-ase-secret-key=*****
2017-01-13 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 /v3/ase/api	Content-Type=application/json, x-ase-access-key=, x-ase-secret-key=*****
GET /v3/ase/api	-SAME AS ABOVE-
DELETE /v3/ase/api	-SAME AS ABOVE-
POST /v3/ase/server	-SAME AS ABOVE-
GET /v3/ase/server	-SAME AS ABOVE-
DELETE /v3/ase/server	-SAME AS ABOVE-
GET /v3/ase/cluster	-SAME AS ABOVE-
POST /v3/ase/firewall	-SAME AS ABOVE-
GET /v3/ase/firewall	-SAME AS ABOVE-
POST /v3/ase/firewall/flowcontrol	-SAME AS ABOVE-
GET /v3/ase/firewall/flowcontrol	-SAME AS ABOVE-
POST /v3/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

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

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](#)
 - [Create an IAM role in the security, identity, and compliance](#)
 - [Create the security group](#)

- [Create launch configuration](#)
- [Create an auto-scale group](#)

Create an AMI for ASE

Complete the following steps to create an AMI for ASE:

1. Create an RHEL 7.2 or later or Ubuntu 16.0 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-aws-rhel-3.2.2.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-aws-rhel-3.2.2.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
RHEL7.2 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

Parent topic: [Autoscaling ASE in AWS environment](#)

Create an IAM role in the security, identity, and compliance

Complete the following steps to create an IAM role in the security, identity, and compliance:

1. Create an IAM role by selecting the EC2 instance:

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

2. Assign **AmazonEC2ReadOnlyAccess** privilege to the role.

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

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](#) 

Cancel Previous **Create role**

3. Provide the role name: * Required

Parent topic: [Autoscaling ASE in AWS environment](#)

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

Parent topic: [Autoscaling ASE in AWS environment](#)

Create launch configuration

Create the launch configuration that the auto-scaling group will use. To create the launch configuration, complete the following 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.

Parent topic: [Autoscaling ASE in AWS environment](#)

Create an auto-scale group

Complete the following steps to create the auto scale group:

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:

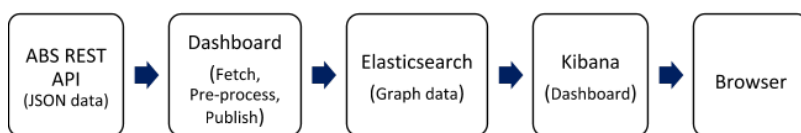
```
/opt/pingidentity/ase/bin$sudo ./uninstall_service.sh
This script will uninstall ASE service
Do you wish to proceed (y/n)? y
Starting service uninstallation
RHEL 7.2 detected, uninstalling ASE Service..
ase stop/waiting
ASE service successfully uninstalled
```

Parent topic:[Autoscaling ASE in AWS environment](#)

PingIntelligence for APIs Dashboard

PingIntelligence for APIs Dashboard, also referred to as Dashboard, utilizes Elasticsearch and Kibana to provide a graphical view of an API environment including traffic metrics, attack information, and blocked connections. The Dashboard makes periodic REST API calls to an ABS engine which returns JSON reports that are used to generate graphs. All the connections between browser to Kibana, from Kibana to Elasticsearch, and from Elasticsearch to Dashboard are secured (SSL) connections. 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 ABS graphs:



For detailed information on installing and configuring an ABS engine, please see the *ABS Admin Guide*.

System requirements for Dashboard

PingIntelligence for APIs Dashboard, Elasticsearch, and Kibana should be installed on a separate Linux server (x86_64 architecture, RHEL 7.2 or higher or Ubuntu 16 LTS). The following table shows the recommended system requirements for up to 10,000 queries per second (QPS) of aggregate traffic.

CPU (2.4 GHz, Intel)	Memory	I/O	Storage
4 CPU	8 GB	1 Gbps	500 GB

The components can be installed in the following environments:

Component	Environment
Kibana	Browser: IE11+, Chrome, Firefox, and Safari (Mac)
Elasticsearch	RHEL 7.2 or higher and Ubuntu 16 LTS
PingIntelligence for APIs Dashboard	RHEL 7.2 or higher and Ubuntu 16 LTS

Dashboard and Elasticsearch can be installed in any of the following environments:

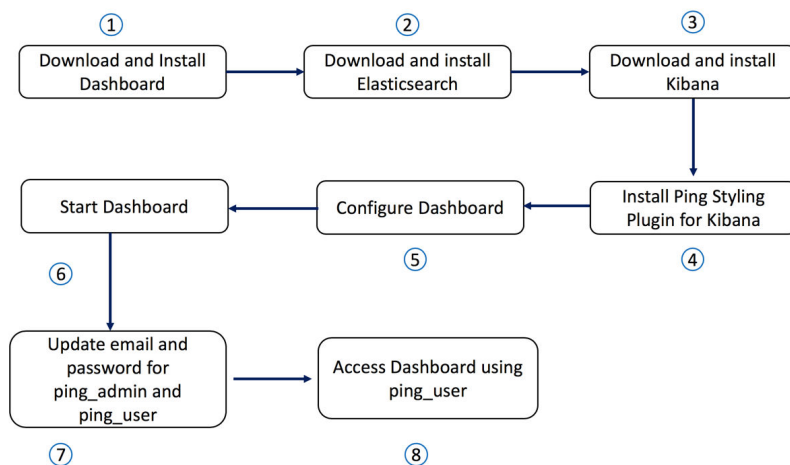
- Amazon EC2
- VMware ESXi
- Bare Metal
- Docker Containers

Install and configure the PingIntelligence for APIs Dashboard

Install the following components on a RHEL 7.2 or higher or Ubuntu 16 LTS Linux Server:

- Elasticsearch 6.4.3
- Kibana 6.4.3
- Oracle JDK 8 update 161 or later

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:

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

Make sure that you create strong passwords of more than six characters for all users. The default password of all user types is `changeme`.

- [Prerequisites to installation and configuration](#)
- [Download and install Dashboard](#)
- [Download and install Elasticsearch](#)
- [Download, install and initialize Kibana](#)
- [Install Ping styling plugin for Kibana](#)
- [Configure Dashboard](#)
- [Update the admin password](#)

Prerequisites to installation and configuration

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

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

Download and install Dashboard

Download Dashboard 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-3.2.1.tar.gz
```

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

Directories	Description
-------------	-------------

bin	ABS Start and Stop scripts; Elasticsearch and Kibana initialization scripts.
config	dashboard.properties file used to configure 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
plugins	Contains the Ping styling plugin
logs	Contains Dashboard log files which are rotated every 24 hours
util	Contains the check_ports_dashboard.sh script to check the availability of default Elasticsearch and ABS ports to connect.

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

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-6.4.3.tar.gz
```



CAUTION: since this command wraps, enter it manually.

7. Install Elasticsearch:

```
$ tar -zxf elasticsearch-6.4.3.tar.gz
```

8. Change directory:

```
$ cd /opt/elasticsearch/elasticsearch-6.4.3
```

- [Configure Elasticsearch](#)

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

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 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 ~]$
```

Parent topic: [Download and install Elasticsearch](#)

Download, install and initialize 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-6.4.3-linux-x86_64.tar.gz"
```

7. Install Kibana:

```
$ tar -zxf kibana-6.4.3-linux-x86_64.tar.gz
```

8. Change directory:

```
$ cd /opt/kibana/kibana-6.4.3-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]
```

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

Install Ping styling plugin for Kibana

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

```
./bin/kibana-plugin install
    file:///opt/pingidentity/dashboard/plugins/pingstyling-3.2.zip
```

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]
```

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

Configure Dashboard


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

```
# Dashboard properties file
# 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+qp6kMrkgNjA
# ABS secret key
abs.secret_key=OBF:AES:NuBmDcAhQeNlPBDmyxX+685CBe8c3/
STVA==:BIfH+FKmL5cNa1DrfVuyc5hIYjimqh7Rnf3bv9hW0+4=
# ABS query polling interval (minutes)
abs.query.interval=10
# ABS query offset (minutes. minimum value 30 minutes)
abs.query.offset=30
# 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=6.4.3
# Log level
dashboard.log.level=INFO
```

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

Parameter	Description
<code>abs.host</code>	<p>IP address of the ABS server</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  Note: 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. </div>
<code>abs.port</code>	<p>REST API port number of the ABS host – See <code>abs.properties</code> Default value is 8080</p>
<code>abs.ssl</code>	<p>Setting the value to true ensures SSL communication between ABS and PingIntelligence for APIs Dashboard</p>
<code>abs.restricted_user</code>	<p>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 ABS users for API reports</p>
<code>abs.access_key</code>	<p>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.</p>
<code>abs.secret_key</code>	<p>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></p>

	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.
<code>abs.query.interval</code>	Polling interval to fetch data from ABS. The default is 10 minutes
<code>abs.query.offset</code>	The time required by ABS to process access logs and generate result. The minimum value is 30 mins and default value is 60 mins.
<code>es.url</code>	Elasticsearch URL
<code>es.username</code>	Elasticsearch username
<code>es.password</code>	Elasticsearch password.
<code>kibana.version</code>	Kibana version - default is 6.4.3
<code>dashboard.log.level</code>	Log level for Dashboard Default log level is INFO . Another log level is DEBUG

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

Update the 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
```

Parent topic: [Install and configure the PingIntelligence for APIs Dashboard](#)

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:

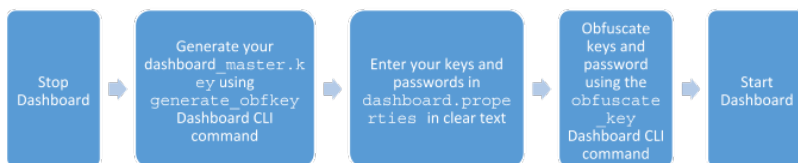
- `abs.access_key`
- `abs.secret_key`
- `es.password`

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

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



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. If you want to restart the Dashboard, the `dashboard_master.key` must be present in the `/opt/pingidentity/dashboard/config/` directory.

Start Dashboard

Prerequisite:

For 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 for security reasons, copy it to the `config` directory before executing the **start** script.

To start the 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 3.2.1 starting...
Please see /opt/pingidentity/dashboard/logs/dashboard.log for more details
[pingidentity@localhost bin]#
```

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

- `ping_admin`
- `ping_user`

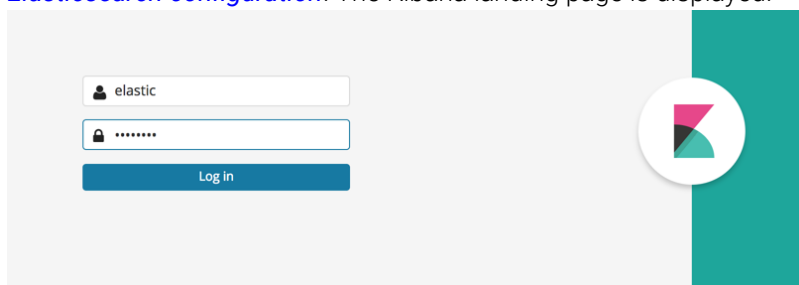


Note: Immediately after starting PingIntelligence for APIs Dashboard, change the password for both the users.

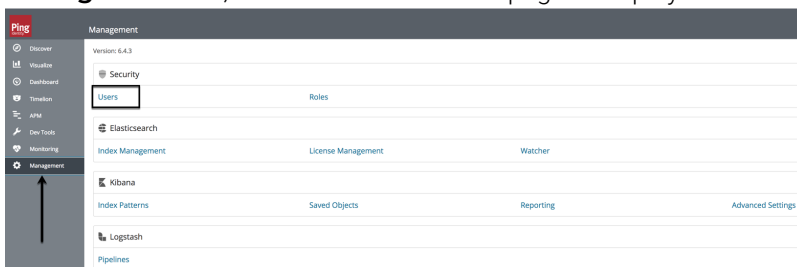
Access the ABS Dashboard

Access the main dashboard with a browser at this URL: `https://<ip:port>/app/kibana#/dashboard/pingapiintelligence`. In the above 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:

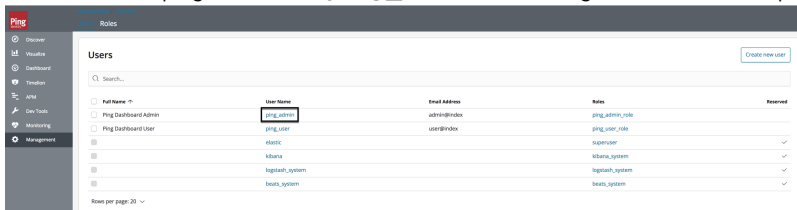
1. Navigate to the ABS Dashboard URL and log in using `elastic` user and the password set during [Elasticsearch configuration](#). The Kibana landing page is displayed.



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



- On the Users page, click on **ping_admin** to change the email and password of ping_admin user.



- 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

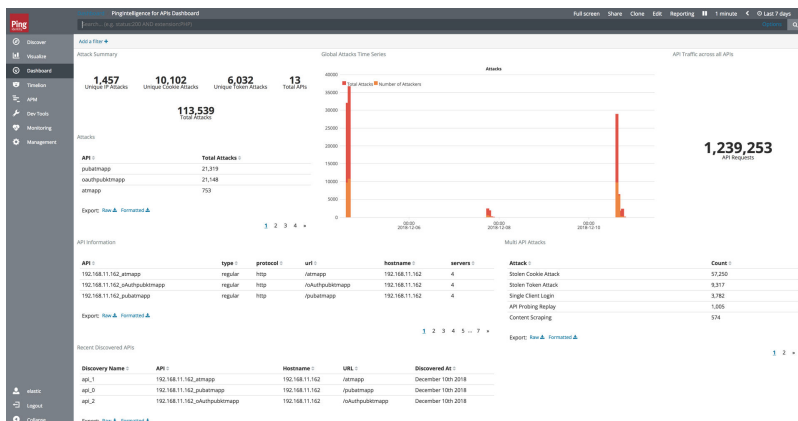
Password

Confirm password

Save password Cancel

Update user Cancel Delete user

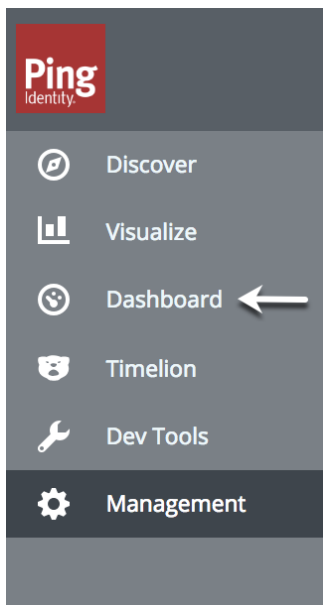
Repeat steps 2 through 4 for ping_user to update **Email** and **Password**. Then log in with ping_user credentials to view the dashboard. Here is a partial screen grab of the main dashboard:



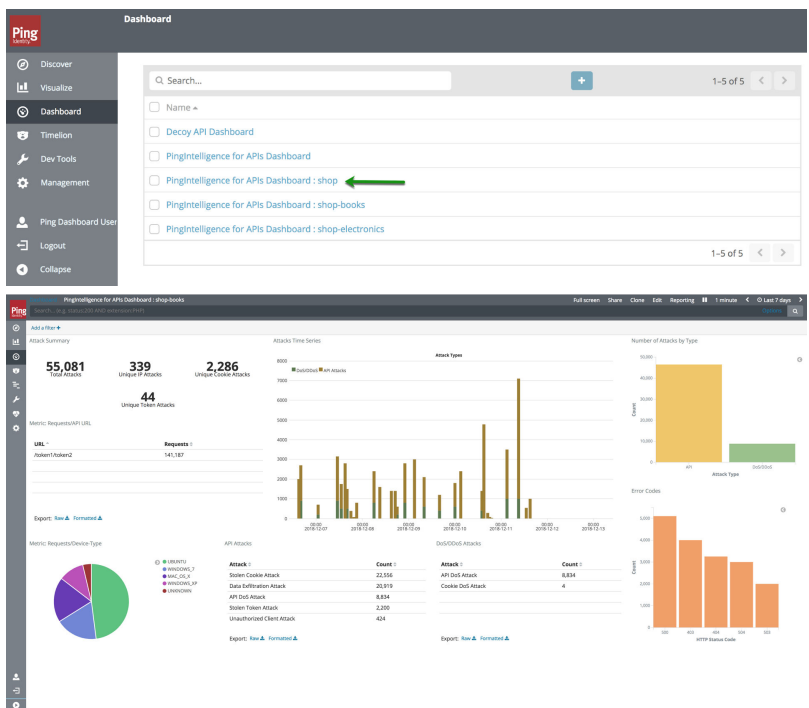
The main dashboard provides the following information:

- **Attack Summary:** total number of attacks, number of unique IP addresses and unique cookies generating attacks. Note: a single IP or cookie could generate more than one attack, so the sum of the unique IPs and cookies 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 API Security Enforcer 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 using the menu on the left-hand side (see graphic to the right). Click **Dashboard** to display the list of APIs for which Dashboards are available.



Click on a listed API name to display the detailed graphs. You can open more than one API by opening each API dashboard in a new tab. A dashboard which is like the one shown below is displayed.



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 unique IP addresses and unique cookies generating attacks. Note: a single IP or cookie could generate more than one attack, so the sum of the unique IPs and cookies 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

Error and traffic control reporting

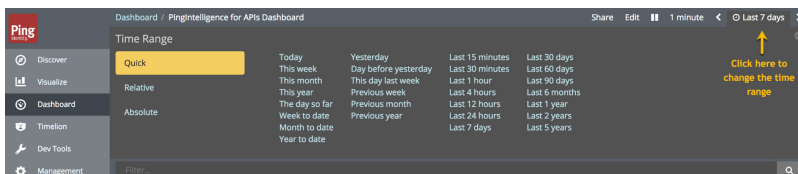
- **Server error codes** – Count of each error code returned from API servers.
- **DoS/DDoS threshold exceeded per API** – Count of traffic thresholds exceeded including Server Spike, Client Spike, and Connection Quota Exceeded (See *API Security Enforcer Admin Guide* for parameters).
- **Blocked connections** – Count of each blocked connection type.

 **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.

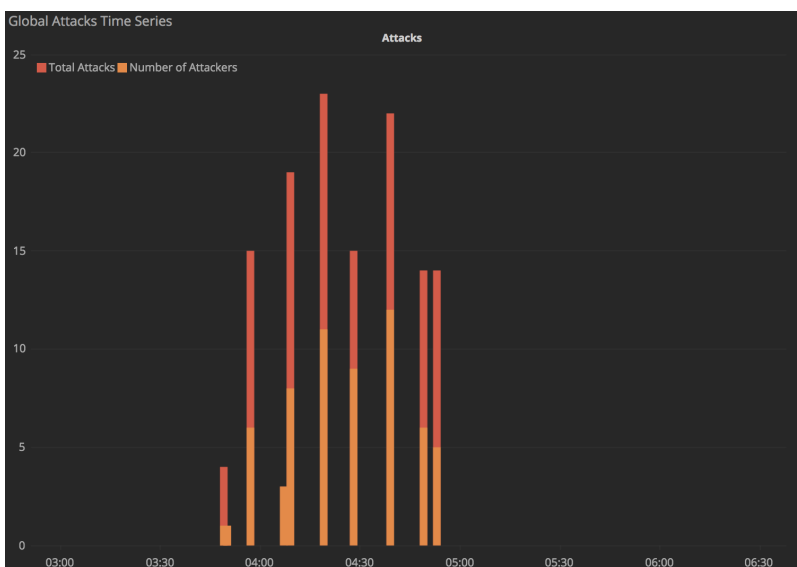
- [Dashboard time series](#)
- [Stop Dashboard](#)

Dashboard time series

ABS 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 as shown in the following screen capture:



Here is a screen capture of the time series data:



Parent topic:[Access the ABS Dashboard](#)

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 3.2.1 stopping...
Dashboard is stopped
```

Parent topic:[Access the ABS Dashboard](#)

Purge data from Elasticsearch

To manage storage on the Dashboard server, you can either archive or purge Elasticsearch data. PingIntelligence provides a purge script to remove older Elasticsearch data.



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

To run the purge script, enter the following in 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 `purge.sh` script either archives or purges processed access log files which are stored in the `/opt/pingidentity/dashboard/logs` directory.



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

Located in the `/opt/pingidentity/dashboard/util` directory, the `purge` script deletes logs older than the specified number of days. Run the script using the ABS 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 `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/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.

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 analytic 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](#)
 - [API Security Enforcer deployment modes](#)
 - [Install ASE software](#)
 - [ASE license](#)
 - [Change default settings](#)
 - [Obfuscate keys and passwords](#)
 - [Start and Stop ASE](#)
 - [Configure SSL for external APIs](#)
 - [ASE cluster setup \(optional\)](#)

ASE ports

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

Port Number	Usage
80	Data port for HTTP and WebSocket connections. Accessible from any client (not secure). If you are installing ASE as a non-root user, choose a port that is greater than or equal to 1024.
443	Data port for HTTPS and Secure WebSocket (wss) connections. Accessible from any client. If you are installing ASE as a non-root user, choose a port that is greater than or equal to 1024.
8010	Management port used by CLI and REST API for managing ASE. Accessible from management systems and administrators
8020	Cluster port used by ASE for cluster communication. Accessible from all cluster nodes.
8080, 9090	ABS ports used by ASE for outbound connections to ABS for sending access logs and receive client identifiers of suspected attacks.



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

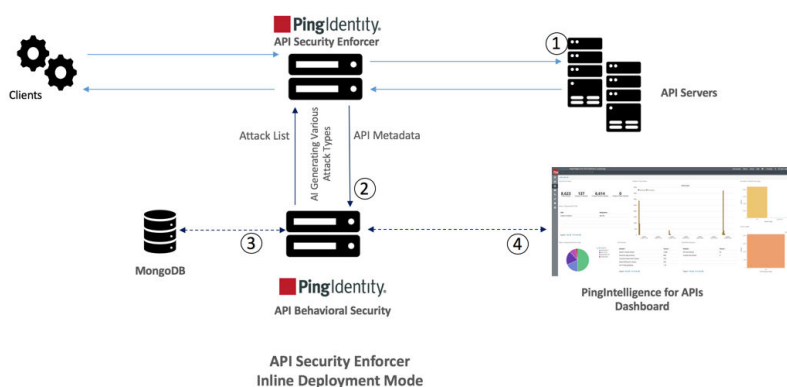
Parent topic: [Part A – Install ASE](#)

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.



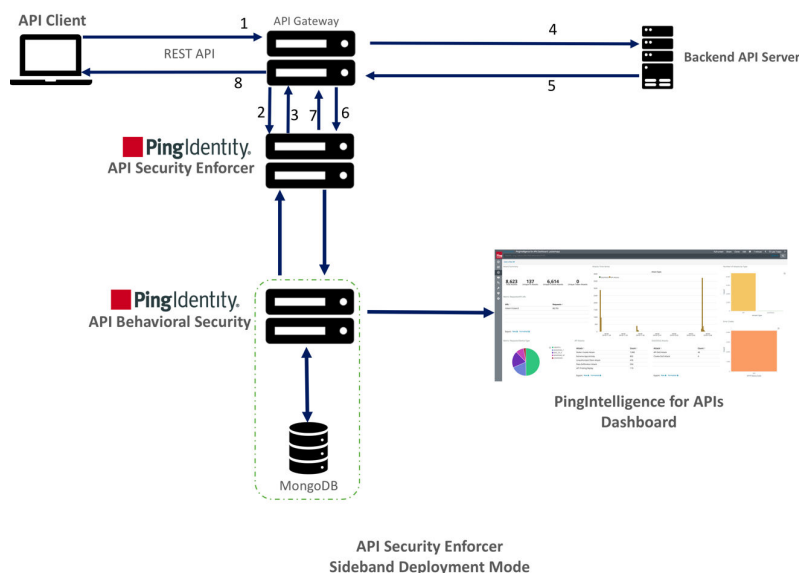
To configure ASE to work in the Inline mode, set the **mode=inline** in the `/opt/pingidentity/ase/config/ase.conf` file.

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.



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](#).

Parent topic: [Part A – Install ASE](#)

Install ASE software

ASE supports RHEL 7.2 or higher or Ubuntu 16 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-aws-rhel-3.2.1.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-aws-rhel-3.2.1.tar.gz
```

Parent topic: [Part A – Install ASE](#)

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.

Parent topic: [Part A – Install ASE](#)

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 3.2.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

Change keystore password

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

Command: `update_keystore_password`

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

Change admin password

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

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

Parent topic: [Part A – Install ASE](#)

Obfuscate keys and passwords

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

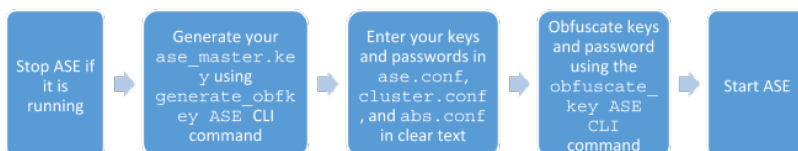
The following keys and passwords are obfuscated in the three configuration files:

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



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

The following diagram summarizes the obfuscation process:



Generate your `ase_master.key`

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

```
/opt/pingidentity/ase/bin/cli.sh generate_obfkey -u admin -p
Please take a backup of config/ase_master.key, config/ase.conf,
config/abs.conf, config/cluster.conf before proceeding
```

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

Warning: Obfuscation master key file `/opt/pingidentity/ase/config/ase_master.key` already exist.

```
This command will delete it create a new key in the same file
Do you want to proceed [y/n]:y
creating new obfuscation master key
Success: created new obfuscation master key at
/opt/pingidentity/ase/config/ase_master.key
```

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



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

Obfuscate key and passwords

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

```
/opt/pingidentity/ase/bin/cli.sh obfuscate_keys -u admin -p
Please take a backup of config/ase_master.key, config/ase.conf, config/
abs.conf, and config/cluster.conf before proceeding
If config keys and password are already obfuscated using the current master
key, it is not obfuscated again
```



```

Following keys will be obfuscated:
config/ase.conf: sender_password, keystore_password
config/abs.conf: access_key, secret_key
config/cluster.conf: cluster_secret_key
Do you want to proceed [y/n]:y
obfuscating config/ase.conf, success
obfuscating config/abs.conf, success
obfuscating config/cluster.conf, success

```

Start ASE after keys and passwords are obfuscated.



Important: After the keys and passwords are obfuscated, the `ase_master.key` must be moved to a secure location from ASE.

Parent topic: [Part A – Install ASE](#)

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.

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

```

/opt/pingidentity/ase/bin/start.sh
Starting API Security Enforcer 3.2.1...
please see /opt/pingidentity/ase/logs/controller.log for more details

```

Stop ASE

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

```

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

```

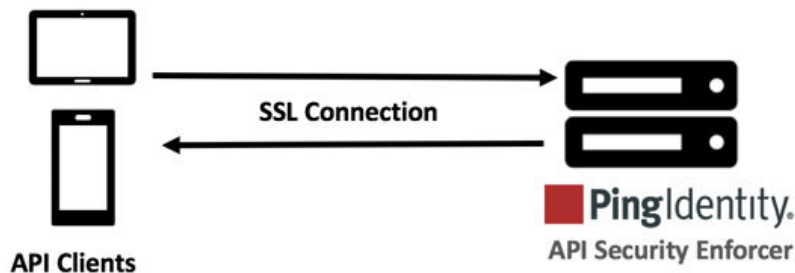
Parent topic: [Part A – Install ASE](#)

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 `/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
```

- 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 `.cert` file.

- Create the key from the existing `.pem` file:

```
openssl rsa -in private.pem -out private.key
```

- Convert the existing `.pem` file to a `.cert` file:

```
openssl x509 -in server-cert.pem -out server-cert.crt
```

- 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
```

- Import the `.cert` file in ASE using the `import_cert` CLI command:

```
/opt/pingidentity/ase/bin/cli.sh import_cert server-cert.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
```

- Restart ASE by stopping and starting.



**Important:** You can also configure for Management APIs. For more information on configuring SSL for management APIs, see [Configure SSL for Management APIs](#).

**Parent topic:** [Part A – Install ASE](#)

### ASE cluster setup (optional)

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



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

To setup an ASE cluster node:

- Navigate to the `config` directory
- Edit `ase.conf` file:
  - 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\_management\_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 down the ASE cluster](#)
- [Delete a cluster node](#)
- [Stop ASE cluster](#)

**Parent topic:** [Part A – Install ASE](#)

## 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
```

Parent topic:[ASE cluster setup \(optional\)](#)

## 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.

Parent topic:[ASE cluster setup \(optional\)](#)

## 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>
```

**Parent topic:** [ASE cluster setup \(optional\)](#)

## 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.

**Parent topic:** [ASE cluster setup \(optional\)](#)

## 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](#)
- [ABS License](#)
- [Obfuscate passwords](#)
- [Configure SSL](#)
- [Import existing CA-signed certificates](#)
- [Install MongoDB software](#)
- [Change default settings](#)
- [Connect ABS to MongoDB](#)
- [Start and Stop ABS](#)

### 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 Oracle JDK 8 update 161 or later on a 64-bit architecture machine with Ubuntu 16 LTS or RHEL7. 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-3.2.1.tar.gz`

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## 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: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.

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## 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`

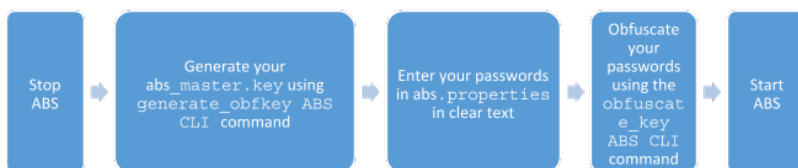


ABS ships with a default `abs_master.key` which is used to obfuscate the various keys and passwords. It is recommended to generate your own `abs_master.key`. The default `jks_passwordabs123` is configured in the `abs.properties` file.



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

The following diagram summarizes the obfuscation process:



### Generate `abs_master.key`

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

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

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

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

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

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

creating new obfuscation master key

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

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



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

### Obfuscate key and passwords

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

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

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

Enter clear text keys and password before obfuscation.

Following keys will be obfuscated

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

```
obfuscating /pingidentity/abs/config/abs.properties
```

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

Start ABS after passwords are obfuscated.



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

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## Configure SSL

ABS supports only TLS 1.2 and requires Java 8 u161 and later. 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"
});
```

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](#) for steps to obfuscate passwords.

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

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## 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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3F
```

Enter the following command:

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

For example:

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

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 3.2.1...
please see /opt/pingidentity/abs/logs/abs/abs.log for more details
```

## Configure ABS system

1. Change your working directory to `/opt/pingidentity/config`
2. Edit `abs.properties` and update the following parameters:

| Parameter                    | Description                                                                                                          |
|------------------------------|----------------------------------------------------------------------------------------------------------------------|
| <code>host_ip</code>         | The externally visible IP address of the ABS system.                                                                 |
| <code>mongo_master_ip</code> | IP address of MongoDB master node.                                                                                   |
| <code>mongo_username</code>  | Match username that is in the <code>abs_init.js</code> file.                                                         |
| <code>mongo_password</code>  | Match password that is in the <code>abs_init.js</code> file. The default password is obfuscated. Change the password |

The following is the snippet of the `abs.properties` file showing the `mongo_username` and `mongo_password`:

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

For more information on `abs.properties` file, see the API Behavioral Admin Guide.

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## Install MongoDB software

ABS uses a MongoDB database (3.4.6) 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 3.4.6 Linux tarball from the MongoDB website. For more information, see <https://www.mongodb.org/downloads>. This document describes a RHEL 7 download, but the equivalent Ubuntu version of MongoDB is also supported.

### 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 3.4.6 on each node and extract to `/opt/pingidentity/mongo`

```
cd /opt/pingidentity/
/opt/pingidentity# wget \
https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-3.4.6.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"
});
```

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.

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## Change default settings

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

### Change default JKS password

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



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

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

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

2. **Change the key password:** Enter the following command to change the key password. The default key password is abs123

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

Start ABS after you have changed the default passwords.

### Change abs\_master.key

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

**Command:** generate\_obfkey. ABS must be stopped before creating a new abs\_master.key

**Stop ABS:** If ABS is running, then stop ABS before generating a new ABS master key. Enter the following command to stop ABS:

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

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

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

### Change CLI admin password

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

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



## Change default access and secret key in MongoDB

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

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>" })
```

**Parent topic:** [Part B – Install ABS and MongoDB](#)

## 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 3.4.6
connecting to: admin
switched to db abs_metadata
WriteResult({ "nInserted" : 1})
bye
```

Parent topic: [Part B – Install ABS and MongoDB](#)

## 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 3.2.1...
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 7.1 (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
```

Parent topic: [Part B – Install ABS and MongoDB](#)

## 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](#)
  - [Enable ASE to ABS engine communication](#)
  - [Add APIs to ASE](#)
  - [Train ABS AI engine](#)

### 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 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 `abs.properties` file (see `abs.properties` in *Obfuscate passwords*).



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

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

Here is a sample `abs.conf` file:

```
; API Security Enforcer ABS configuration.
; This file is in the standard .ini format. The comments start with a
; semicolon (;).
; Following configurations are applicable only if ABS is enabled with true.

; a comma-separated list of abs nodes having hostname:port or ipv4:port as
; an address.
abs_endpoint=127.0.0.1:8080

; access key for abs node
access_key=OBF:AES://
ENOzsqOEhDBWLDY+pIoQ:jN6wLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0
```

```

; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw:zzi77GIFF5xkQJccjIrIVWU+RY5CxUhp3NLcNBel+3Q

; Setting this value to true will enable encrypted communication with ABS.
enable_ssl=true

; Configure the location of ABS's trusted CA certificates. If empty, ABS's
certificate
; will not be verified
abs_ca_cert_path=

```



**Important:** Make sure that ASE and ABS are in the same time zone.

**Parent topic:** [Part C – Integrate ASE and ABS](#)

## 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 : disabled, 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

```

**Parent topic:** [Part C – Integrate ASE and ABS](#)

## Add APIs to ASE

After installing ASE and ABS Engine, the next step is to add API definitions to the PingIntelligence for APIs software. This process can be completed automatically or manually.

### Automatic API discovery

ABS AI Engine supports automatic discovery of APIs. The ABS Engine Admin Manual API Discovery section explains this process which operates as follows:

- When traffic from an unknown API is passed to the ASE, it forwards the traffic metadata to the AI Engine which automatically discovers the API definition.
- Ping Identity Automated API Definition (AAD) tool will then generate an API JSON file and load it to the ASE system.

After the API JSON definition is loaded, the AI Engine begins the training process. See the "Training" chapter in the ABS Admin Guide for more information on training the AI model.

### Manual configuration of API definitions

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 ASE Admin Guide.

1. Navigate to `/opt/pingidentity/api_proxy/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                                                                                                                                                                                                                                                                               |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>protocol</b> | API type:<br>http<br>- HTTP /REST API,<br>ws<br>- WebSocket                                                                                                                                                                                                                               |
| <b>url</b>      | The value of the URL for the managed API. You can configure up to three levels of sub-paths. For example,<br>"/shopping"-<br>name of a 1 level API<br>"/shopping/electronics/<br>phones" -<br>3 level API<br>"/"<br>-<br><br>entire server (used for ABS API Discovery or load balancing) |
| <b>hostname</b> | HTTP host header, for example,<br>"api.xyz.com"<br>The value cannot be empty.<br>"*"<br>matches any hostname.                                                                                                                                                                             |
| <b>cookie</b>   | Name of cookie used by backend servers.                                                                                                                                                                                                                                                   |

| Parameter                                                                                                                                                | Description                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>oauth2_access_token</b>                                                                                                                               | When <code>true</code> , ASE captures OAuth2 Access Tokens.                                                                                                              |
| <b>apikey_qs</b>                                                                                                                                         | When API Key is sent in the query string, ASE uses the specified <code>parameter name</code> to capture the API key value.                                               |
| <b>apikey_header</b>                                                                                                                                     | When API Key is part of the header field, ASE uses the specified <code>parameter name</code> to capture the API key value.                                               |
| <b>login_url</b>                                                                                                                                         | Public URL used by a client to connect to the application.                                                                                                               |
| <b>server_ssl</b>                                                                                                                                        | When <b>true</b> , ASE uses SSL/TLS to secure backend connection. Default value is <code>false</code> .                                                                  |
| Servers:<br><b>host</b><br><b>port</b>                                                                                                                   | For each backend server running the API, configure: <ul style="list-style-type: none"> <li>• Host - IP address or hostname</li> <li>• Port - the port number.</li> </ul> |
| Flow Control:<br><b>client_spike_threshold</b><br><b>server_connection_queueing</b><br><b>bytes_in_threshold (WS)</b><br><b>bytes_out_threshold (WS)</b> | ASE Flow Control ensures that backend API servers are protected from unplanned or malicious (for example, DDoS) surges in API traffic.                                   |
| <b>protocol_allowed</b>                                                                                                                                  | Accepted protocols - HTTP, HTTPS, WS, or WSS.                                                                                                                            |
| <b>methods_allowed</b>                                                                                                                                   | Accepted methods. GET, POST, PUT, DELETE, HEAD                                                                                                                           |
| <b>content_type_allowed</b>                                                                                                                              | Allowed content types allowed. For example, <code>application/json</code>                                                                                                |

| Parameter                                                                            | Description                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Decoy Config:<br><b>decoy_enabled</b><br><b>response_code</b><br><b>response_def</b> | When <b>decoy_enabled</b> is set to <b>true</b> , configured decoy sub-paths work as decoy APIs.<br><br><b>response_code</b> is the status code (for example, 200 ) that ASE returns when a decoy API path is accessed. |

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
```

### Check the availability of access logs in ABS

Navigate to the data directory and check whether it has access logs sent by API Security Enforcer.

### Enable attack blocking (optional)

ABS Engine generates a list of clients which executed attacks on an API service and can be configured to automatically send the attack list to ASE which blocks client access. By default, automatic blocking is inactive. Execute the following to activate automatic client blocking:

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

**Parent topic:** [Part C – Integrate ASE and ABS](#)

## 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"
 });

```

### 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 **training\_period** 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>/v3/abs/admin`. Following is a snippet of the output of the Admin API:

```

"message": "training started at Thu Aug 09 12:32:59 IST 2018",
"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.

Parent topic: [Part C – Integrate ASE and ABS](#)

## Part D – Install PingIntelligence for APIs Dashboard

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](#)
  - [Change Dashboard default settings](#)
  - [Obfuscate keys and passwords](#)
  - [Install Elasticsearch](#)
  - [Install Kibana](#)
  - [Install Ping styling plugin for 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. 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 PingIntelligence for APIs Dashboard

```
$ tar -zxf dashboard-3.2.1.tar.gz
```

The following table shows the directories created when PingIntelligence for APIs 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 PingIntelligence for APIs Dashboard<br>A subdirectory called <code>dashboard</code> containing Kibana schema for each API |
| <code>data</code>   | Temporary storage for ABS data                                                                                                                                                     |
| <code>lib</code>    | Contains <code>dashboard.jar</code> and dependent external jar files                                                                                                               |
| <code>logs</code>   | Contains PingIntelligence for APIs 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.                                             |

**Parent topic:** [Part D – Install PingIntelligence for APIs Dashboard](#)

## Change Dashboard default settings

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

1. Log in to the management `t2.micro` instance
2. Change directory to `software`
3. Untar Dashboard binary:

```
tar -zxvf dashboard-3.2.1.tar.gz
```

### 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-3.2.1 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.

**Parent topic:** [Part D – Install PingIntelligence for APIs Dashboard](#)

## 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:

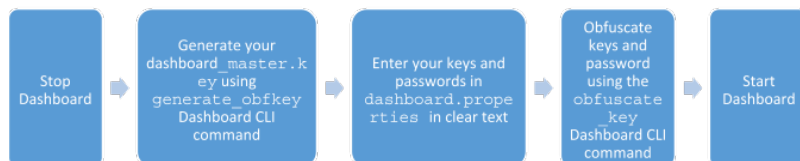
- `abs.access_key`
- `abs.secret_key`
- `es.password`

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
```



**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.

**Parent topic:** [Part D – Install PingIntelligence for APIs 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-6.4.3.tar.gz
```



**CAUTION:** since this command wraps, enter it manually.

7. Install Elasticsearch:

```
$ tar -zxf elasticsearch-6.4.3.tar.gz
```

8. Change directory:

```
$ cd /opt/elasticsearch/elasticsearch-6.4.3
```

- [Configure Elasticsearch](#)

**Parent topic:** [Part D – Install PingIntelligence for APIs Dashboard](#)

## 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 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 ~]$
```

Parent topic: [Install Elasticsearch](#)

## 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-6.4.3-linux-x86_64.tar.gz"`



**CAUTION:** since this command wraps, enter it manually.

7. Install Kibana:

```
$ tar -zxvf kibana-6.4.3-linux-x86_64.tar.gz
```

8. Change directory:

```
$ cd /opt/kibana/kibana-6.4.3-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]
```

Parent topic:[Part D – Install PingIntelligence for APIs Dashboard](#)

## Install Ping styling plugin for Kibana

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

```
./bin/kibana-plugin install
 file:///opt/pingidentity/dashboard/plugins/pingstyling-3.2.zip
```

Parent topic:[Part D – Install PingIntelligence for APIs Dashboard](#)


## Integrate dashboard with ABS AI engine

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](#)) 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 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+qp6kMrkgNjA
ABS secret key
abs.secret_key=OBF:AES:NuBmDcAhQeNlPBDmyxX+685CBe8c3/
STVA==:BIFh+FKmL5cNa1DrfVuyC5hIYjimqh7Rnf3bv9hW0+4=
ABS query polling interval (minutes)
abs.query.interval=10
ABS query offset (minutes. minimum value 30 minutes)
abs.query.offset=30
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/
RLUN5rbvZLTPghqVZzD9V:+ZGHbhpY4HENYYqJ4wn50AmoO6CZ30cfjqTYQCfgBgc=
kibana version
kibana.version=6.4.3
Log level
dashboard.log.level=INFO
```

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

| Parameter                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>abs.host</code>            | <p>IP address of the ABS server</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <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. </div>                        |
| <code>abs.port</code>            | <p>REST API port number of the ABS host – See <code>abs.properties</code><br/>Default value is 8080</p>                                                                                                                                                                                                                                                                                                                             |
| <code>abs.ssl</code>             | <p>Setting the value to true ensures SSL communication between ABS and PingIntelligence for APIs Dashboard</p>                                                                                                                                                                                                                                                                                                                      |
| <code>abs.restricted_user</code> | <p>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></p> |
| <code>abs.access_key</code>      | <p>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.</p>                     |
| <code>abs.secret_key</code>      | <p>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.</p>                     |
| <code>abs.query.interval</code>  | <p>Polling interval to fetch data from ABS. The default is 10 minutes</p>                                                                                                                                                                                                                                                                                                                                                           |
| <code>abs.query.offset</code>    | <p>The time required by ABS to process access logs and generate result. The minimum value is 30 mins and default value is 60 mins.</p>                                                                                                                                                                                                                                                                                              |
| <code>es.url</code>              | <p>Elasticsearch URL</p>                                                                                                                                                                                                                                                                                                                                                                                                            |



|                            |                                                                                            |
|----------------------------|--------------------------------------------------------------------------------------------|
| <b>es.username</b>         | Elasticsearch username                                                                     |
| <b>es.password</b>         | Elasticsearch password.                                                                    |
| <b>kibana.version</b>      | Kibana version - default is 6.4.3                                                          |
| <b>dashboard.log.level</b> | Log level for Dashboard<br>Default log level is<br>INFO<br>. Another log level is<br>DEBUG |

**Parent topic:** [Part D – Install PingIntelligence for APIs Dashboard](#)

## 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
```

```
[pingidentity@localhost bin]# ./start.sh
Dashboard 3.2.1 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

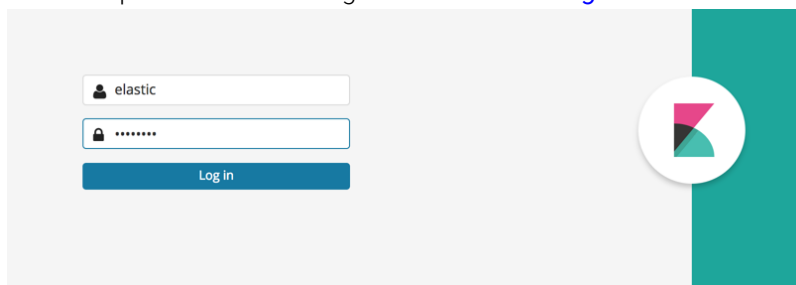


**Note:** Immediately after starting PingIntelligence for APIs Dashboard, change the password for both the users.

## Connect to the dashboard

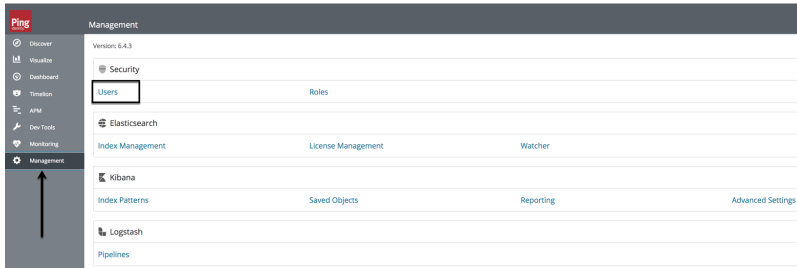
Access <https://<ip:port>/app/kibana#/dashboard/pingapiintelligence> to load the main dashboard. In the above link, `<ip:port>` is the IP address and port (default – 443) configured in `kibana.yml`. Change the password of users `ping_admin` and `ping_user` by completing the following steps:

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

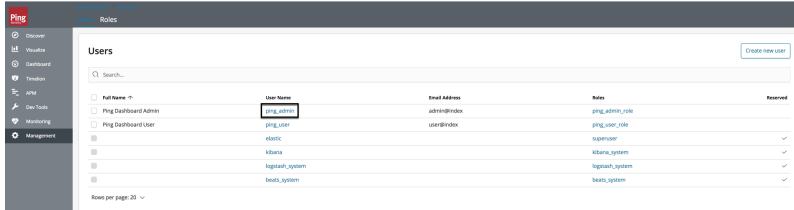


landing page is displayed.

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



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



- On the **ping\_admin Users** page, update the **Email** and **Password** 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

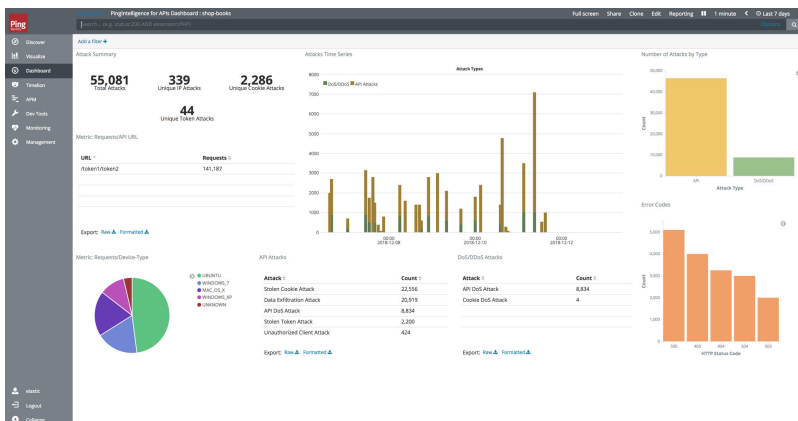
Password  
\*\*\*\*\*

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:



Parent topic: [Part D – Install PingIntelligence for APIs 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](#)
- [Using ABS self-signed certificate with Postman](#)
- [View ABS Reports in Postman](#)

### 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.

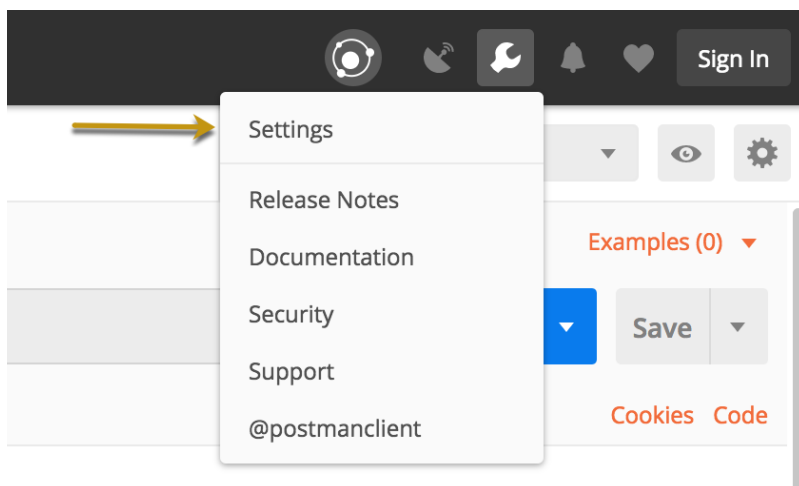
Parent topic: [Part E – Access ABS reporting](#)

### Using ABS self-signed certificate with Postman

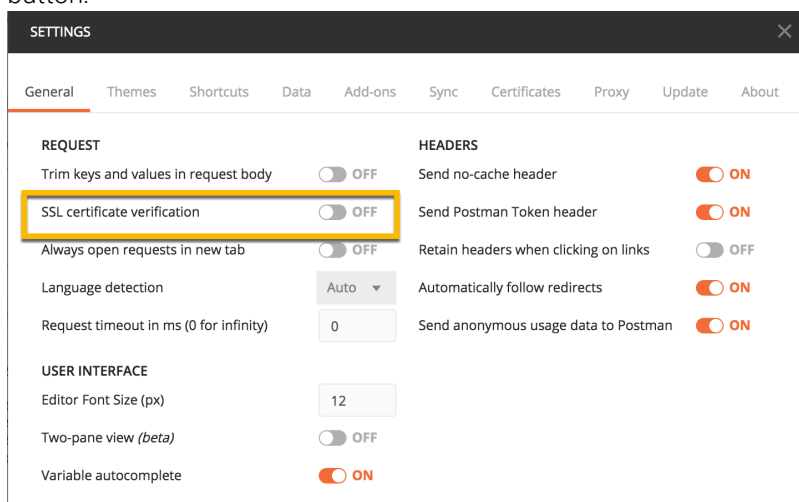
ABS ships with a self-signed certificate. If you want to use Postman with the self-signed certificate of ABS, then from Postman's settings, disable the certificate verification option. Complete the following steps to disable Postman from certificate verification:



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:

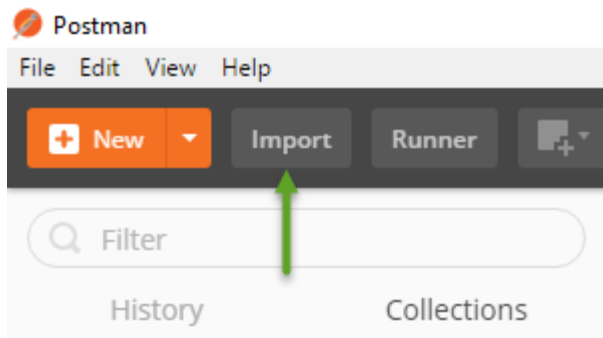


Parent topic: [Part E – Access ABS reporting](#)

## View ABS Reports in Postman

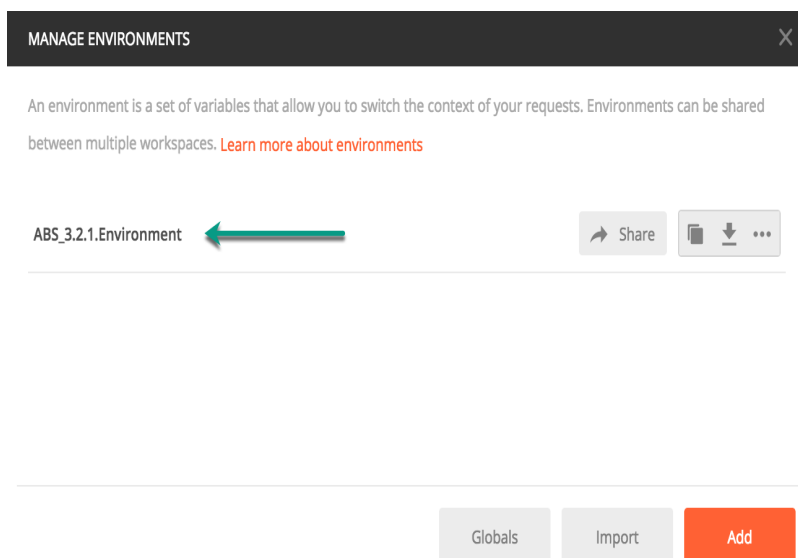
To view the reports, complete the following steps:

1. Download `ABS_3.2.1_Environment` and `ABS_3.2.1_Reports` JSON files from **API Reports Using Postman** folder on Ping Identity [Download](#) site. These configuration files will be used by Postman.
2. [Download](#) and install the Postman application 6.2.5 or higher.
3. In Postman, **import** the two Ping Identity files downloaded in step 1 by clicking the Import button.



4. After importing the files, click the gear  button in the upper right corner.

5. In the **MANAGE ENVIRONMENTS** pop-up window, click **ABS\_3.2.1\_Environment**



6. In the pop-up window, configure the following values and then click **Update**

- **Server:** IP address of the ABS node for which the `dashboard_node` was set to `true` in the [abs.properties](#) file.
- **Port:** Port number of the ABS node.
- **Access\_Key\_Header** and **Secret\_Key\_Header:** Use the Admin user or Restricted user header. A Restricted user sees obfuscated value of OAuth token, cookie and API keys. For more information of different types of user, see [ABS users for API reports](#)
- **Access\_Key** and **Secret\_Key:** The Access Key and Secret Key configured in the `opt/pingidentity/mongo/abs_init.js` for either admin or restricted user. Make sure that access key and secret key corresponds to the admin or restricted user header configured.
- **API\_Name:** The name of the API for which you want to generate the reports.
- **Later\_Date:** A date which is more recent in time. For example, if the query range is between March 12 and March 14, then the later date would be March 14.
- **Earlier\_Date:** A date which is past in time. For example, if the query range is between March 12 and March 14, then the earlier date would be March 12.



**Note:** Do not edit any fields that start with the word `system`.

MANAGE ENVIRONMENTS
✕

Environment Name

ABS\_3.2.1.Environment

|                                     | VARIABLE              | INITIAL VALUE             | CURRENT VALUE                                            |  |  |
|-------------------------------------|-----------------------|---------------------------|----------------------------------------------------------|--|--|
| <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              | PingIntelligenceApp       | PingIntelligenceApp                                      |  |  |
| <input checked="" type="checkbox"/> | Later_Date            | 2019-03-31T18:00          | 2019-03-31T18:00                                         |  |  |
| <input checked="" type="checkbox"/> | Earlier_Date          | 2019-01-31T13:30          | 2019-01-31T13:30                                         |  |  |
| <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. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#)
✕

Cancel Update

- 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.

Parent topic: [Part E – Access ABS reporting](#)

## 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:

- *PingAccess integration*
- *Axway integration*
- *Apigee integration*

## Deployment in AWS environment

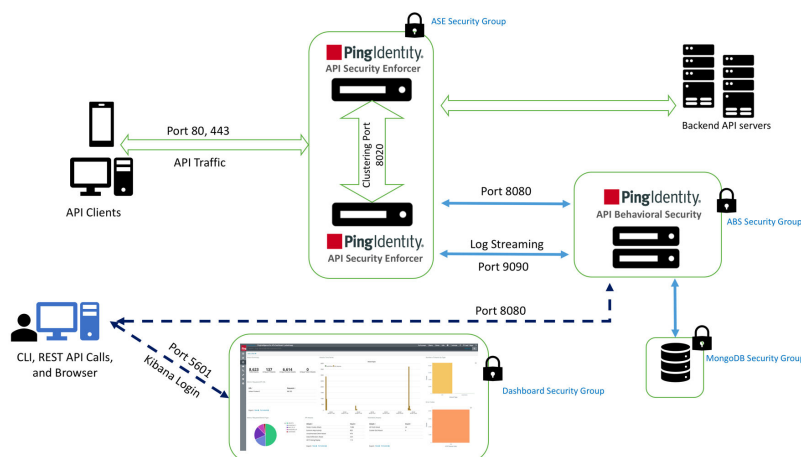
### Automated AWS setup

This Ansible package automates the deployment of PingIntelligence for APIs software in an AWS environment. The package installs and configures four software components which include ASE (deployed between the API clients and API Gateway or backend server), ABS AI Engine, MongoDB database, and the PingIntelligence for APIs dashboard. Using the automated PingIntelligence deployment script, you can choose from one of the following deployment types:

- **Single instance deployment** - In this type of deployment, all the PingIntelligence components are deployed on a single EC2 instance.

- **Separate server deployment** - In this type of deployment, all the PingIntelligence components are deployed on different EC2 instances.

The following diagram shows the complete deployment architecture of the setup:



### Important:

PingIntelligence automated deployment script creates AWS EC2 instances and installs the PingIntelligence software. The installation is divided in the following steps:

1. In [step 1](#), download and untar the automated installation package. Configure the `aws.config` script that governs the installation. It is recommended to run the automation script as a `ec2-user`. The script installs packages like Ansible and boto using `sudo` on the management EC2 instance.
2. (Optional): Change the default passwords, keys, and port numbers
3. In [step 2](#), launch the EC2 instances.
4. In [step 3](#), configure few system parameters for PingIntelligence components.
5. In [step 4](#), install the PingIntelligence components.

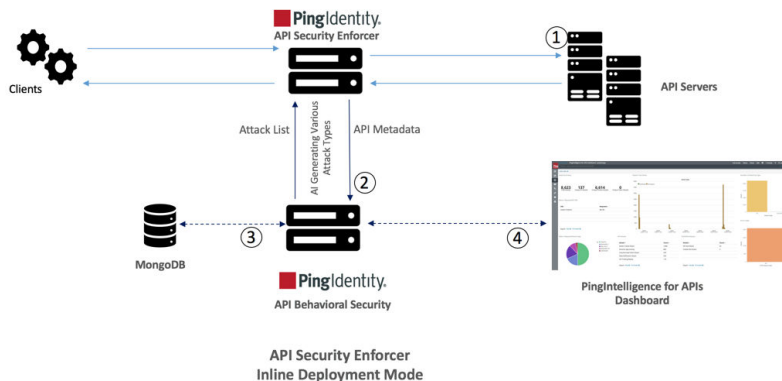
- [ASE deployment modes](#)
- [Step 1: Set up the host system](#)
- [Change ASE's default settings](#)
- [Change ABS default settings](#)
- [Change Dashboard default settings](#)
- [Step 2: Launch EC2 instances](#)
- [Step 3: Configure system parameters](#)
- [Step 4: Install PingIntelligence for APIs software](#)
- [Verify the PingIntelligence installation](#)
- [Next steps - Integrate PingIntelligence into your environment](#)
- [Shutdown the deployment](#)
- [View installation logs](#)

## 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 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.

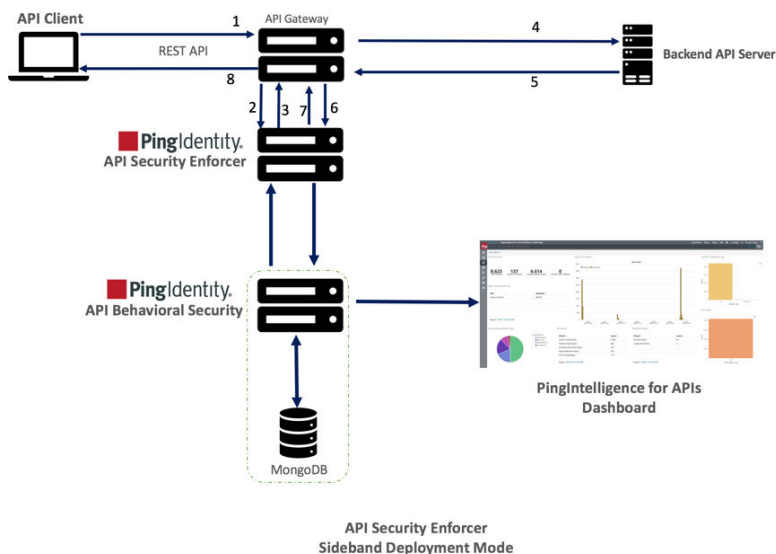


To configure ASE to work in the Inline mode, set the **mode=inline** in the `aws.config` file.

### 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 Inline mode, set the **mode=sideband** in the `aws.config` 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 [PingIdentity documentation site](#).

**Parent topic:** [Automated AWS setup](#)

## Step 1: Set up the host system

Step 1 of the installation is divided in following three parts:

- Download and untar the automation package
- Configure the `aws.config` file
- Copy the ASE and ABS license in the respective `license` directory
- Download PingIntelligence and third-party software


### Download and untar the automation package

1. Create a `t2.micro` RHEL 7.3 instance and login to the server. The Ansible script creates the setup in the same Amazon region and VPC subnet as the `t2.micro` instance.
2. [Download](#) the AWS Ansible file from the PingIntelligence section and, for example, save in the `/home/ec2-user` directory.
3. Untar the downloaded file:
 

```
$ cd /home/ec2-user
$ tar -xf aws-production-3.2.2.tar.gz
```

Untarring the file creates the following subdirectories in the `pingidentity/aws-production` directory:

| Directory            | Description                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------|
| <code>ansible</code> | Contains the different <code>yml</code> files                                                                    |
| <code>bin</code>     | Contains the <code>start.sh</code> and <code>stop.sh</code> scripts. Do not edit the contents of this directory. |
| <code>certs</code>   | The <code>certs</code> directory contains ASE, ABS, Elasticsearch, and Kibana self-signed certificates and keys. |

 **Note:**

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | <ul style="list-style-type: none"> <li>• If you want to use your 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.</li> <li>• If you are replacing the default certificates and keys, make sure that the keys do not have a password.</li> <li>• The certificates should be in <code>pem</code> format.</li> </ul> |
| <code>config</code>   | <p>Contains the following configuration and default settings files:</p> <ul style="list-style-type: none"> <li>• <code>aws.config</code></li> <li>• <code>abs-defaults.yml</code></li> <li>• <code>ase-defaults.yml</code></li> <li>• <code>dashboard-defaults.yml</code></li> </ul>                                                                                                                                                                     |
| <code>data</code>     | Contains the MongoDB key file and SSH key                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>external</code> | The third-party components like MongoDB are downloaded in the <code>external</code> directory.                                                                                                                                                                                                                                                                                                                                                           |
| <code>keys</code>     | Contains master key of ASE, ABS, and Dashboard. Automated installation generates new master key for each PingIntelligence component. These keys are populated in this directory at the end of the installation process.                                                                                                                                                                                                                                  |
| <code>license</code>  | Contains <code>ase</code> and <code>abs</code> directories that have the ASE and ABS license file.                                                                                                                                                                                                                                                                                                                                                       |
| <code>logs</code>     | Contains the log files for automated installation                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>software</code> | <p>Contains the binary files for PingIntelligence components:</p> <ul style="list-style-type: none"> <li>• ASE</li> <li>• ABS</li> <li>• Dashboard</li> </ul>                                                                                                                                                                                                                                                                                            |

- [ASE and ABS license](#)
- [Configure aws.config file](#)
- [Manually download third-party components](#)
- [Download PingIntelligence software](#)

Parent topic:[Automated AWS setup](#)

## 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=3.2
IssueDate=2018-11-30
EnforcementType=0
ExpirationDate=2018-12-30
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=3.2
IssueDate=2018-11-30
EnforcementType=0
ExpirationDate=2018-12-30
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/ase/PingIntelligence.lic
Module=ABS
```




Parent topic:[Step 1: Set up the host system](#)

## Configure `aws.config` file

Complete the following steps to configure the `aws.config` file. This file controls the automation script to install PingIntelligence software in an AWS environment.

Navigate to the `aws-production/config` directory and edit the `aws.config` file. The following table describes the various variables of the `aws.config` file. The configuration file has parameters where link to download third-party component is configured. If your `t3.micro` instance, does not have internet access, download the [third-party components manually](#).

| Variable                                                                                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• <code>AWS_ACCESS_KEY</code></li> <li>• <code>AWS_SECRET_KEY</code></li> </ul> | The access and secret key of the AWS account                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>installation</code>                                                                                              | <p>Configure the type of installation. Setting the value to <code>single</code> installs all the following components on a single EC2 instance:</p> <ul style="list-style-type: none"> <li>• API Security Enforcer (ASE)</li> <li>• API Behavioral System (ABS)</li> <li>• PingIntelligence for APIs Dashboard</li> <li>• MongoDB</li> </ul> <p>Setting the value to <code>separate</code>, installs each component on a separate EC2 instance. The default value is <code>single</code>.</p>                                                                                                                                                                                                                              |
| <code>installation_path</code>                                                                                         | <p>Configure the path where you would want the PingIntelligence components to be installed. The default value is <code>/home/ec2-user</code>.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> <b>Important:</b> The path that you provide in the <code>installation_path</code> variable must exist on the EC2 instance. The automation script does not create this path. If you have chosen <code>installation</code> as <code>separate</code> in the preceding variable, manually create the same path on each EC2 instance before running the <a href="#">PingIntelligence setup step</a>.</p> </div> |
| <code>install_with_sudo</code>                                                                                         | <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.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

|                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>jdk8_download_url</p>                                                                                                                                                                                                                                                     | <p>The automated script requires Oracle JDK8 u161 and later. Complete the following steps to download and save JDK8:</p> <ol style="list-style-type: none"> <li>Download and save Linux_x64 Oracle JDK 8 update 161 rpm or later in the external directory.</li> </ol> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p> <b>Note:</b> Make sure to download the tar.gz file and not the rpm file.</p> </div> <ol style="list-style-type: none"> <li>Rename the downloaded JDK 8 rpm file to jdk8.tar.gz</li> </ol> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p> <b>Note:</b> Make sure the JDK 8 download URL has not expired.</p> </div> |
| <p>mongodb_download_url</p>                                                                                                                                                                                                                                                  | <p>MongoDB download URL. A default URL is populated in the aws.config file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <p>elasticsearch_download_url</p>                                                                                                                                                                                                                                            | <p>Elasticsearch download URL. A default URL is populated in the aws.config file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <p>kibana_download_url</p>                                                                                                                                                                                                                                                   | <p>Kibana download URL. A default URL is populated in the aws.config file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <p>EC2 instance types for a separate deployment type:</p> <ul style="list-style-type: none"> <li>ase_instance_type - ASE</li> <li>abs_instance_type - ABS</li> <li>mongo_instance_size - MongoDB</li> <li>dashboard_instance_type - Dashboard</li> <li>abs_report</li> </ul> | <p>The following are the default instance sizes:</p> <ul style="list-style-type: none"> <li>ASE - m4.2xlarge</li> <li>ABS - m4.4xlarge</li> <li>MongoDB - m4.2xlarge</li> <li>Dashboard - m4.2xlarge</li> <li>ABS reporting node - m4.2xlarge</li> </ul> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p> <b>Note:</b> If abs_reporting_instance_type is left empty, then no separate ABS reporting node is created. Dashboard connects to the first ABS node.</p> </div>                                                                                                                                                                                                                                                                           |

|                                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>ing_instance_type - ABS reporting node</pre>                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                  |
| <p>Volume size for different instance type for a separate deployment:</p> <ul style="list-style-type: none"> <li>• <pre>ase_volume_size - ASE</pre></li> <li>• <pre>abs_volume_size - ABS</pre></li> <li>• <pre>mongo_volume_size - MongoDB</pre></li> <li>• <pre>dashboard_volume_size - Dashboard</pre></li> <li>• <pre>abs_reporting_volume_size - ABS reporting node</pre></li> </ul> | <p>The following are the default volume sizes:</p> <ul style="list-style-type: none"> <li>• ASE - 1000 GB</li> <li>• ABS - 1000 GB</li> <li>• MongoDB - 1000 GB</li> <li>• Dashboard - 1000 GB</li> <li>• ABS reporting node - 250 GB</li> </ul> |
| <p>Number of instances. Applicable for separate deployment type.</p> <ul style="list-style-type: none"> <li>• <pre>ase_instances - ASE</pre></li> <li>• <pre>abs_instances - ABS</pre></li> <li>• <pre>mongo_instances</pre></li> </ul>                                                                                                                                                   | <p>The default number of instances:</p> <ul style="list-style-type: none"> <li>• ASE - 1</li> <li>• ABS - 1</li> <li>• MongoDB - 1</li> </ul>                                                                                                    |

|                                                     |                                                                         |
|-----------------------------------------------------|-------------------------------------------------------------------------|
| - MongoDB                                           |                                                                         |
| Instance type for single deployment - instance_type | The default value for a single instance deployment is m4.4xlarge.       |
| Volume size for single deployment - volume_size     | The default value for volume for single instance deployment is 1000 GB. |

Following is a sample `aws.config` file:

```
[all:vars]

AWS keys
AWS_ACCESS_KEY=
AWS_SECRET_KEY=

Installation type. Valid values are
1) single - single consolidated server for all PingIntelligence components
2) separate - separate servers for each PingIntelligence component
installation=single

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

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

Download URLs for external packages
jdk8_download_url='jdk8-download-url'

mongodb_download_url='https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-
rhel70-3.4.6.tgz'
elasticsearch_download_url='https://artifacts.elastic.co/downloads/
elasticsearch/elasticsearch-6.4.3.tar.gz'
kibana_download_url='https://artifacts.elastic.co/downloads/kibana/
kibana-6.4.3-linux-x86_64.tar.gz'

Configure EC2 Instance types(Valid for separate servers for each
PingIntelligence component)
ase_instance_type=m4.2xlarge
abs_instance_type=m4.4xlarge
mongo_instance_type=m4.2xlarge
dashboard_instance_type=m4.2xlarge
abs_reporting_instance_type=m4.2xlarge
```

```
Configure volume size of instances in GBs(Valid for separate servers for
each PingIntelligence component)
ase_volume_size=1000
abs_volume_size=1000
mongo_volume_size=1000
dashboard_volume_size=1000
abs_reporting_volume_size=250

Configure number of instances for ASE and ABS(Valid for separate servers
for each PingIntelligence component)
ase_instances=1
abs_instances=1
mongo_instances=1

#Configure instance type for a single consolidated server installation
instance_type=m4.4xlarge

#Configure volume for a single consolidated server installation
volume_size=1000
```

Parent topic:[Step 1: Set up the host system](#)

## Manually download third-party components

If the **t2.micro** server does not have internet access, follow the steps below. Download the individual components, save the files in the `external` directory.

1. Install Ansible version 2.6.2 on the `t2.micro` RHEL 7.3 instance.
2. Download the following packages and copy to the external directory using the specified names:
  - **MongoDB** – Download MongoDB from: [https://fastdl.mongodb.org/linux/mongodb-linux-x86\\_64-rhel70-3.4.6.tgz](https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-3.4.6.tgz) and save the file in the external directory as `mongodb.tgz`.
  - **Elasticsearch** – Download Elasticsearch from: <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.4.3.tar.gz> and save the file in the external directory as `elasticsearch-6.4.3.tar.gz`.
  - **Kibana** – Download Kibana from: [https://artifacts.elastic.co/downloads/kibana/kibana-6.4.3-linux-x86\\_64.tar.gz](https://artifacts.elastic.co/downloads/kibana/kibana-6.4.3-linux-x86_64.tar.gz) and save the file in the external directory as `kibana-6.4.3-linux-x86_64.tar.gz`.

Parent topic:[Step 1: Set up the host system](#)

## Download PingIntelligence software

**Download** the following PingIntelligence for APIs software to the `aws-production/software` directory

- API Security Enforcer (AWS RHEL7)
- API Behavioral Security
- PingIntelligence Dashboard



**Note:** Do not change the downloaded file names.

The software directory should include the following files:

```
-rw-r--r--. 1 ec2-user ec2-user 2.5M Dec 05 00:01 dashboard-3.2.1.tar.gz
-rw-r--r--. 1 ec2-user ec2-user 159M Dec 05 00:01 abs-3.2.2.tar.gz
```



```
-rw-r--r--. 1 ec2-user ec2-user 38M Dec 05 00:01 ase-aws-
rhel-3.2.2.tar.gz
```

Parent topic: [Step 1: Set up the host system](#)

## 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> . |
| 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> </ul>                                 |

|                    |                                                                                                                                                                                                                                                                                                                                                  |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | <ul style="list-style-type: none"> <li>• <code>sender_email</code><br/>: Email address used from which email alerts and reports are sent.</li> <li>• <code>email_password</code><br/>: Password of sender's email account.</li> <li>• <code>receiver_email</code><br/>: 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.                                                                                                                                                                                                                           |



**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

 # 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
```

**Parent topic:** [Automated AWS setup](#)

## 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><br>and<br><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>                                     | Default and maximum value is 40 for a multi-instance deployment. If you are deploying all the components on a single EC2 instance, this value is internally set to 25.                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>access_key</code><br>and<br><code>secret_key</code>         | The access key and secret for the admin user. For more information on different ABS users, see <a href="#">ABS users</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>access_key_ru</code><br>and<br><code>secret_key_ru</code>   | The access key and secret for the restricted user. For more information on different ABS users, see <a href="#">ABS users</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>jks_password</code>                                         | 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><br/>: Set it to <code>true</code> for ASE to send email notifications. Default value is <code>false</code>.</li> <li>• <code>smtp_host</code><br/>and<br/><code>smtp_port</code></li> <li>• <code>sender_email</code><br/>: Email address used from which email alerts and reports are sent.</li> <li>• <code>email_password</code><br/>: Password of sender's email account.</li> <li>• <code>receiver_email</code><br/>: 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. |
|--|------------------------------------------------------------------|



**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: 50

 # 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



```

Parent topic: [Automated AWS setup](#)

## Change Dashboard default settings

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

| Variable | Description |
|----------|-------------|
|----------|-------------|

|                     |                                                                                                                                                                                                                                                                                                                           |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kibana_port         | Port used to access the PingIntelligence Dashboard.                                                                                                                                                                                                                                                                       |
| jks_password        | The password of the JKS Keystore. The default password is abs123.                                                                                                                                                                                                                                                         |
| elastic_password    | Elasticsearch password. The default value is changeme.<br><br><div style="border: 1px solid black; padding: 5px;">  <b>Note:</b> Do not change the <code>elastic_password</code> after PingIntelligence installation is complete. </div> |
| kibana_password     | Kibana password. The default value is changeme.<br><br><div style="border: 1px solid black; padding: 5px;">  <b>Note:</b> Do not change the <code>kibana_password</code> after PingIntelligence installation is complete. </div>         |
| ping_user_password  | Password for the default user name <code>ping_user</code> . The default value is changeme,                                                                                                                                                                                                                                |
| ping_admin_password | Password for the admin. The default value is changeme.                                                                                                                                                                                                                                                                    |
| CLI admin password  | The default value for CLI admin is <code>admin</code> . To change the password, you need to know the current password.                                                                                                                                                                                                    |



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

 # Passwords for elasticsearch, kibana, ping_user, and ping_admin users
 # 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

```

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

Parent topic: [Automated AWS setup](#)

## Step 2: Launch EC2 instances

Run the following command to launch the EC2 instances:

```
[ec2-user@ip-172-31-8-101 aws-production]$./bin/start.sh launch
Please see /home/ec2-user/pingidentity/aws-production/logs/ansible.log for
more details.
```



**Note:** Do not execute the `run.sh` script available in the `bin` directory.

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

```
[ec2-user@ip-172-31-8-101 aws-production]$ tail -f logs/ansible.log
=====
Current Time: Thu Feb 14 05:35:36 UTC 2019
Starting launch scripts
=====
Thu Feb 14 05:35:36 UTC 2019: Setting up local environment
Thu Feb 14 05:35:36 UTC 2019: Installing packages

PLAY [Provision EC2 Instances]

TASK [Launch of EC2 instances successful]

PLAY RECAP

127.0.0.1 : ok=30 changed=22 unreachable=0 failed=0

Thu Feb 14 05:37:36 UTC 2019: EC2 instances and security groups created
successfully
=====
```

At the end of end of launch step, a `hosts` file is generated which has the IP addresses of all EC2 instances. A snippet of a sample `config/hosts` file is shown below:

```
[ase]
172.31.0.19 id=i-0f342f32f92396425 public_ip=18.191.14.16

[elasticsearch]
172.31.15.183 id=i-01e154407b5bd0836 public_ip=18.188.93.156

[dashboard]
172.31.15.183 id=i-01e154407b5bd0836 public_ip=18.188.93.156
```

```
[kibana]
172.31.15.183 id=i-01e154407b5bd0836 public_ip=18.188.93.156

[abs]
172.31.2.82 id=i-069d4f0d85a4a504c public_ip=18.219.99.40

[abs_reporting_node]
172.31.12.182 id=i-069d4f1d85a5b604d public_ip=18.219.98.20

[mongodb]
172.31.10.227 id=i-094c16d68fd03dd29 public_ip=18.220.185.142

[all:vars]
ansible_ssh_private_key_file=/home/ec2-user/pingidentity/aws-production/data/
pingid_apisecurity_production_key.pem
ansible_ssh_user=ec2-user
```

At the end of the automated deployment, the PingIntelligence software are installed on a single EC2 instance or on multiple EC2 instances based on the deployment type you configured in the `aws.config` file. In order to access the EC2 instances, use the `ansible_ssh_private_key_file` available in the `hosts` file.

- [PingIntelligence AWS instance and security group name](#)

**Parent topic:** [Automated AWS setup](#)

## AWS instance and security group name

**Instance names:** Following are the instance names of the EC2 instances created during the deployment:

- `pingid_apisecurity_production_ase`
- `pingid_apisecurity_production_abs`
- `pingid_apisecurity_production_dashboard`
- `pingid_apisecurity_production_abs_reporting_node`
- `pingid_apisecurity_production_mongodb`

**Security group names:** AWS security groups are also built as part of the setup. Make sure that you do not have any other instances or security groups with the same names.

| Component | Security Group                                          |
|-----------|---------------------------------------------------------|
| ASE       | <code>pingid_apisecurity_production_ase_sg</code>       |
| ABS       | <code>pingid_apisecurity_production_abs_sg</code>       |
| Dashboard | <code>pingid_apisecurity_production_dashboard_sg</code> |
| MongoDB   | <code>pingid_apisecurity_production_mongodb_sg</code>   |

At the end of running of automated deployment, the master keys of all the products are deleted from the individual instances for security reasons. The master keys are available only in the `tar` file of the software

package on the management instance. Starting of any PingIntelligence component requires a master key to be present in the config directory. To manually restart any of the PingIntelligence components, copy the corresponding master key back to the `config` directory. For more information on master key of each component, see the respective *Admin Guide*.

**Parent topic:** [Step 2: Launch EC2 instances](#)

### Step 3: Configure system parameters

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

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

Run the following command to configure the system parameters on the EC2 respective instances:

```
[ec2-user@ip-172-31-8-101 aws-production]$./bin/start.sh configure
Please see /home/ec2-user/pingidentity/aws-production/logs/ansible.log for
more details.
```

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

```
[ec2-user@ip-172-31-8-101 aws-production]$ tail -f logs/ansible.log
=====
Current Time: Thu Feb 14 05:57:18 UTC 2019
Starting configure scripts
=====
Successfully installed pip-19.0.2
Thu Feb 14 05:57:24 UTC 2019: Play configure scripts

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 [Configure system settings for ASE] *****

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 *****
172.31.2.34 : ok=16 changed=6 unreachable=0 failed=0

Thu Feb 14 05:57:52 UTC 2019: Configure successful
=====
```

**Parent topic:** [Automated AWS setup](#)

### Step 4: Install PingIntelligence for APIs software

Run the following command to install the PingIntelligence components



```
[ec2-user@ip-172-31-8-101 aws-production]$./bin/start.sh install
Please see /home/ec2-user/pingidentity/aws-production/logs/ansible.log for
more details.
```



**Note:** Do not execute the `run.sh` script available in the `bin` directory.

An example `ansible.log` file for a successful setup is shown below:

```
[ec2-user@ip-172-31-8-101 aws-production]$ tail -f logs/ansible.log
=====
Current Time: Fri Feb 15 04:43:55 UTC 2019
Starting setup scripts
=====
Fri Feb 15 04:43:55 UTC 2019: Setting up local environment
Fri Feb 15 04:43:55 UTC 2019: Installing packages
LAY RECAP

127.0.0.1 : ok=7 changed=0 unreachable=0 failed=0
172.31.2.34 : ok=126 changed=79 unreachable=0 failed=0

Fri Feb 15 04:54:49 UTC 2019: Setup successful
=====
```

At the end of the automated deployment, the PingIntelligence software are installed on a single EC2 instance or on multiple EC2 instances based on the deployment type you configured in the [aws.config](#) file.

### Updated PingIntelligence packages

The automated deployment framework creates the updated package for each PingIntelligence component and stores them in the `/pingidentity/aws-production/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 `/pingidentity/aws-production/config` directory. You can use these packages to install PingIntelligence components on other instances.

**Parent topic:** [Automated AWS setup](#)

## Verify the installation

Verify that all the components have installed and started successfully. Login to individual instances using the management instance (`t2.micro`). The default security groups created for each component allow access only on port number 22 by the management instance. The username is `ec2-user` and the SSH key is available in `data/pingid_apisecurity_prodcution_key.pem`

### Verify ASE installation

Log in to the ASE EC2 instance and navigate to `/home/ec2-user/pingidentity/ase/bin` directory and run the `status` command:

```
/home/ec2-user/pingidentity/ase/bin/cli.sh status
Ping Identity Inc., API Security Enforcer
status : started
```

```

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>/v3/abs/admin>. Use the IP address from the hosts file generated at the end of the [deployment](#).

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": [
],
 "abs_cluster": {
 "abs_nodes": [
 {
 "node_ip": "192.168.11.166",
 "os": "Red Hat Enterprise Linux Server release 7.3 (Maipo)",
 "cpu": "16",
 "memory": "64G",
 "filesystem": "29%",
 "bootup_date": "Tue Nov 20 16:16:56 IST 2018"
 }
],
 },
 "mongodb_nodes": [
 {
 "node_ip": "192.168.11.168",
 "status": "up"
 },
 {

```

```

 "node_ip": " 192.168.11.169",
 "status": "up"
 },
 {
 "node_ip": " 192.168.11.171",
 "status": "up"
 }
]
},
"ase_logs": [
]
}

```

## 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.

**Parent topic:** [Automated AWS setup](#)

## Next steps - Integrate PingIntelligence into your environment

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

**Sideband configuration:** If you have 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](#)

**Inline configuration:** If you configured PingIntelligence ASE for [inline connectivity](#), 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 ports that you configured in the [aws.config](#) file.

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 based on the deployment mode:
  - [Inline ASE](#), or
  - [Sideband ASE](#)
- [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 generated in the `config/hosts` file.
  - Port number: 8080
  - 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>:5601/>. In the above URL, Kibana IP is the IP address of the Kibana EC2 instance generated in `config/hosts` file.

Login to PingIntelligence for APIs Dashboard using the `ping_user` login ID and the default password `changeme`. Change the default password after you login. The PingIntelligence for APIs Dashboard takes approximately one hour to start showing attack information.

Parent topic:[Automated AWS setup](#)

## Shutdown the deployment

To shut down the deployment and remove all EC2 instances and data, run the `stop.sh` command.



**Note:** When you shut down the deployment, all the EC2 instances along with the data is deleted.

```
[ec2-user@ip-172-31-8-101 aws-production]$./bin/stop.sh
Please see /home/ec2-user/pingidentity/aws-production/logs/ansible.log for
more details.
```

A snippet of the `ansible.log` file for a successful shutdown is shown below:

```
[ec2-user@ip-172-31-8-101 aws-production]$ tail -f logs/ansible.log
=====
Current Time: Wed Feb 20 05:37:36 UTC 2019
Starting stop scripts
=====
Wed Feb 20 05:41:36 UTC 2019: Play terminate scripts
.
.
PLAY RECAP

127.0.0.1 : ok=20 changed=16 unreachable=0 failed=0
Wed Feb 20 05:45:36 UTC 2019: Stop successful
=====
```

Parent topic:[Automated AWS setup](#)

## View installation logs

The `ansible.log` file is available in the `/opt/pingidentity/aws-production/logs` directory.

Parent topic:[Automated AWS setup](#)

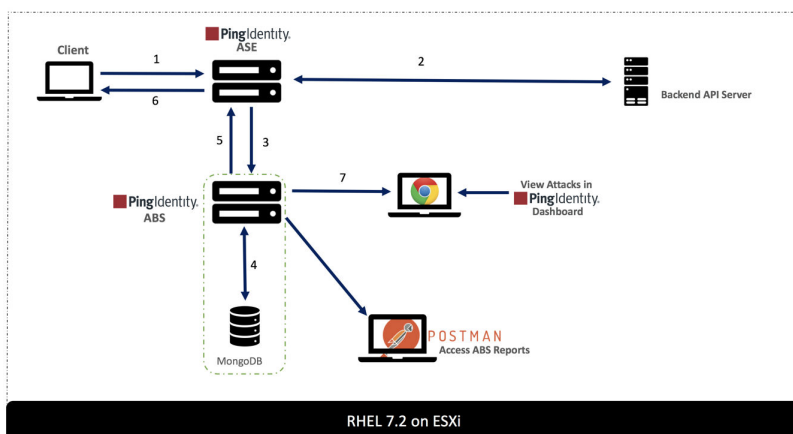
## Deployment on VMs

---

### Automated VM setup

This guide describes the installation and execution of an Ansible package which automatically builds a VMWare ESXi environment with PingIntelligence for APIs software. The package installs and configures four software components which include ASE (deployed between the API clients and API Gateway or backend server), ABS AI Engine, MongoDB database, and the PingIntelligence for APIs dashboard. The entire setup is deployed in an ESXi environment running RHEL 7.2 or later.

The following diagram shows the complete deployment architecture of the setup:

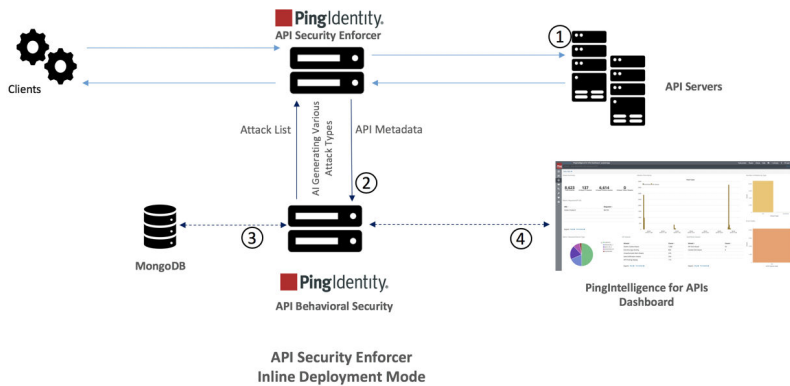


- [ASE deployment modes](#)
- [Step 1: Setup host system](#)
- [Change ASE's default settings](#)
- [Change ABS's default settings](#)
- [Change Dashboard default settings](#)
- [Step 2: Configure system parameters](#)
- [Step 3: Install the PingIntelligence for APIs software](#)
- [Verify PingIntelligence Installation](#)
- [Restart PingIntelligence components \(optional\)](#)
- [Next steps - Integrate PingIntelligence into your environment](#)
- [Shut down the deployment](#)
- [Logs](#)

## 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.

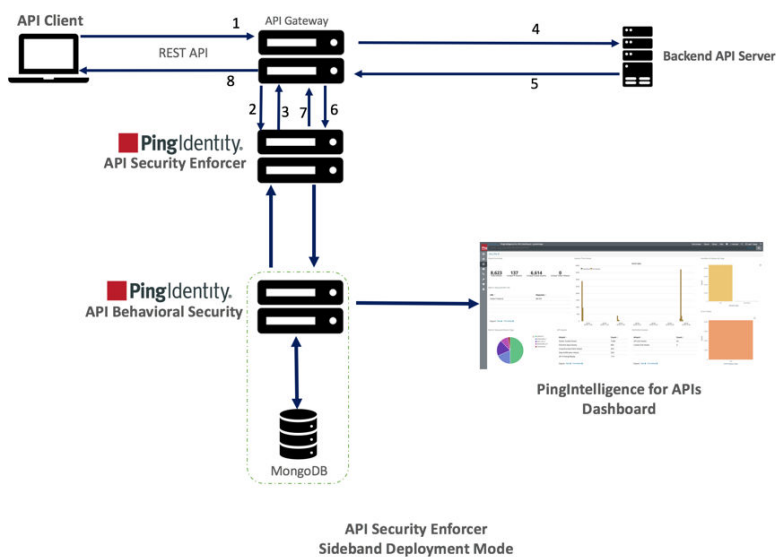


To configure ASE to work in the Inline mode, set the **mode=inline** in the `aws.config` file.

### 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 `aws.config` 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 documentaion site](#).

Parent topic:[Automated VM setup](#)

## Step 1: Setup host system

### Setup the management VM

Create RHEL 7.2 or higher VMs based on the sizing requirements. All the VMs must have internet connectivity to download the third-party tools. Create the VMs based on the sizing decisions based on your API environment.

#### Steps for all the VMs

Complete the following steps on all the VMs except the `ping-management` VM. `ping-management` is the VM from which the ansible script is run to deploy the various PingIntelligence software.



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

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

```
#useradd vm-user
```

2. Change the password

```
#passwd vm-user
```

3. Add the user to the wheel group

```
#usermod -aG wheel vm-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 ping-management VM

1. Login to VM as a root user.
2. [Download](#) the ESXi Ansible file and save it to the `/opt` directory
3. Untar the downloaded file:

```
#tar -xf /opt/esxi-production-3.2.1.tar.gz
```

Untarring the file creates the following subdirectories in the `esxi-production` directory:

| Directory             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ansible</code>  | Contains the different yml files                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>bin</code>      | Contains the <code>start.sh</code> and <code>stop.sh</code> scripts. Do not edit the contents of this directory.                                                                                                                                                                                                                                                                                                                                                      |
| <code>certs</code>    | Contains Elasticsearch and Kibana self-signed certificates and keys. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <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. </div> |
| <code>config</code>   | Contains the <code>aws.config</code> file. This file is used to configure the various variables for installing PingIntelligence components.                                                                                                                                                                                                                                                                                                                           |
| <code>data</code>     | Contains the MongoDB key file                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>external</code> | The third-party components like MongoDB are downloaded in the <code>external</code> directory.                                                                                                                                                                                                                                                                                                                                                                        |
| <code>license</code>  | Contains <code>ase</code> and <code>abs</code> directories that have the ASE and ABS license file.                                                                                                                                                                                                                                                                                                                                                                    |
| <code>logs</code>     | Contains the log files for automated installation                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>software</code> | Contains the binary files for PingIntelligence components: <ul style="list-style-type: none"> <li>• ASE</li> <li>• ABS</li> <li>• Dashboard</li> </ul>                                                                                                                                                                                                                                                                                                                |

- [Configure ASE and ABS license](#)
- [Configure hosts file](#)
- [Manually download third-party components](#)
- [Download PingIntelligence software](#)



- [Secure PingIntelligence Components](#)

Parent topic:[Automated VM setup](#)

## 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=3.2
IssueDate=2018-11-30
EnforcementType=0
ExpirationDate=2018-12-30
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=3.2
IssueDate=2018-11-30
EnforcementType=0
ExpirationDate=2018-12-30
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/ase/PingIntelligence.lic
Module=ABS
```

Parent topic:[Step 1: Setup host system](#)

## 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 Ping Management VM does not have internet access, download the [third-party components manually](#).

1. Navigate to the `esxi-production/config` directory and edit the `hosts` file to add the IP address of all the VMs. For more information see **Configure the hosts file** title below.
2. Run the following command on the Ping Management VM. The Ping Management VM is the VM 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.

3. Run the following command to add the ssh key:

```
#ssh-add
```


4. Run the following command for each VM except the Ping Management VM:




```
ssh-copy-id vm-user@<ping-VM IPv4 address>
```



For example, `ssh-copy-id vm-user@192.168.11.101 (ping-ase)`

## 5. 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>•</li> <li>  [elasticsearch]</li> <li>• [kibana]</li> <li>•</li> <li>  [abs_reporting_node]</li> </ul> | Configure the IP addresses of ASE, ABS, Dashboard, Elasticsearch, Kibana, and MongoDB and ABS reporting node. If you want to install all the components on a single VM, give the same IP address for all the component. If you want a distributed deployment, provide different IP addresses.                                                                             |
| mode                                                                                                                                                                                                                                     | The mode in which you want to deploy API Security Enforcer (ASE). Set the value to <code>inline</code> to deploy ASE in inline mode. Set the value to <code>sideband</code> to deploy ASE in sideband mode. The default value is <code>inline</code>                                                                                                                      |
| installation_path                                                                                                                                                                                                                        | Configure the path where you would want the PingIntelligence components to be installed. The default value is <code>/home/vm-user</code> . <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <b>Important:</b> The path that you provide in the           </div> |

|                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                | <p><code>installation_path</code> variable must exist on the VM. The automation script does not create this path. If you are installing all the PingIntelligence components on different VMs, then manually create the same path on each VM before running the automation script.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Ports                          | <p>Configure the following ports:</p> <ul style="list-style-type: none"> <li>• <code>ase_http_ws_port</code></li> <li>• <code>ase_https_wss_port</code></li> <li>• <code>ase_management_port</code></li> <li>• <code>ase_cluster_manager_port</code></li> <li>• <code>abs_management_port</code></li> <li>• <code>abs_log_port</code></li> <li>• <code>kibana_port</code></li> </ul> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> <b>Note:</b> If you are installing the PingIntelligence components without root access then make sure that the ports numbers assigned are greater than or equal to 1024.</p> </div>                                                                                                                                                                                                                                                          |
| <code>install_with_sudo</code> | <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.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>jdk8_download_url</code> | <p>The automated script requires Oracle JDK8 <code>u161</code> and later. Complete the following steps to download and save JDK8:</p> <ol style="list-style-type: none"> <li>a. Download and save <code>Linux_x64 Oracle JDK 8 update 161</code> or later in the <code>external</code> directory. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> <b>Note:</b> Make sure to download the <code>tar.gz</code> file and <b>not the rpm</b> file.</p> </div> </li> <li>b. Rename the downloaded JDK 8 file to <code>jdk8.tar.gz</code>.</li> </ol> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> <b>Note:</b> Do not edit the <code>jdk8_download_url</code>. The automated deployment uses the JDK 8 downloaded and saved in the <code>external</code></p> </div> |

|                             |                                                                                                                                                                                                                                                                                                               |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | directory in step a.                                                                                                                                                                                                                                                                                          |
| mongodb_download_url        | MongoDB download URL. A default URL is populated in the <code>hosts</code> file.                                                                                                                                                                                                                              |
| elasticsearch_download_url  | Elasticsearch download URL. A default URL is populated in the <code>hosts</code> file.                                                                                                                                                                                                                        |
| kibana_download_url         | Kibana download URL. A default URL is populated in the <code>hosts</code> file.                                                                                                                                                                                                                               |
| elastic_password            | <p>Password to access the elastic user. You can change the default password.</p> <div style="border: 1px solid black; padding: 5px;">  <b>Note:</b> Change the default password before running the deployment script. </div> |
| kibana_password             | <p>Password for Kibana. You can change the default password.</p> <div style="border: 1px solid black; padding: 5px;">  <b>Note:</b> Change the default password before running the deployment script. </div>                 |
| mongo_username              | The default username for MongoDB                                                                                                                                                                                                                                                                              |
| mongo_password              | The default password for <code>mongo_username</code> .                                                                                                                                                                                                                                                        |
| abs_reporting_instance_type | ABS reporting node. This is the node that is used by PingIntelligence Dashboard to fetch data for displaying on the Dashboard.                                                                                                                                                                                |
| ansible_ssh_user            | Ansible <code>ssh</code> user. The default value is <code>esxi-user</code> .                                                                                                                                                                                                                                  |

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

[all:vars]

Deployment mode for ASE. Valid values are inline or sideband
mode=inline

Installation Path
installation_path="/home/esxi-user"

Define ports for the Pingintelligence components
Make sure ports are not same for the same host
ase_http_ws_port=8090
ase_https_wss_port=8443
ase_management_port=8010
ase_cluster_manager_port=8020
abs_management_port=8080
abs_log_port=9090
kibana_port=5601

Configure install_with_sudo to true if the any of the ports in above
list
is <1024. That component will be started using sudo.
install_with_sudo=false

Download URLs for external packages

jdk8_download_url='jdk8-download-url'

mongodb_download_url='https://fastdl.mongodb.org/linux/mongodb-linux-
x86_64-rhel70-3.4.6.tgz'
elasticsearch_download_url='https://artifacts.elastic.co/downloads/
elasticsearch/elasticsearch-6.4.3.tar.gz'
kibana_download_url='https://artifacts.elastic.co/downloads/kibana/
kibana-6.4.3-linux-x86_64.tar.gz'

Passwords for elasticsearch and kibana users
Dashboard will be accessible by these accounts
Please set strong passwords
elastic_password=elAstIc@123!
kibana_password=KibAnA@678!

Mongo DB User and password
mongo_username=absuser
mongo_password=abs123

```

```
#Ansible SSH user with password-less access
ansible_ssh_user=esxi-user
```

Parent topic:[Step 1: Setup host system](#)

## 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 from: [https://fastdl.mongodb.org/linux/mongodb-linux-x86\\_64-rhel70-3.4.6.tgz](https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-3.4.6.tgz) and save the file in the `external` directory as `mongodb.tgz`.

**Elasticsearch** – Download Elasticsearch from: <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.4.3.tar.gz> and save the file in the `external` directory as `elasticsearch-6.4.3.tar.gz`.

**Kibana** – Download from: [https://artifacts.elastic.co/downloads/kibana/kibana-6.4.3-linux-x86\\_64.tar.gz](https://artifacts.elastic.co/downloads/kibana/kibana-6.4.3-linux-x86_64.tar.gz) and save the file in the `external` directory as `kibana-6.4.3-linux-x86_64.tar.gz`.

Parent topic:[Step 1: Setup host system](#)

## Download PingIntelligence software

**Download** the following PingIntelligence for APIs software to `esxi-production/software` directory.

- API Security Enforcer (ESXi RHEL 7)
- API Behavioral Security
- PingIntelligence Dashboard



**Note:** Do not change the name of the downloaded files.

The software directory should include the following files:

```
-rw-r--r--. 1 pingidentity pingidentity 2.5M Dec 07 00:01
dashboard-3.2.1.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 159M Dec 07 00:01 abs-3.2.1.tar.gz
-rw-r--r--. 1 pingidentity pingidentity 38M Dec 07 00:01 ase-esxi-
rhel-3.2.1.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 only in the tar file of the software package 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*.

**Parent topic:** [Step 1: Setup host system](#)

## Secure Components

After you have downloaded the four PingIntelligence for APIs components, each PingIntelligence for APIs software component has a master key (for example, ASE uses `ase_master.key`) which is used to secure keys and passwords. Although a default master key is provided, PingIntelligence recommends generating your own secure master key. Here is the process to follow to secure each software component.

**ASE:** ASE uses the master key defined in `ase_master.key` to obfuscate keys and passwords. Detailed information on generating the master key and securing content is available in the [Obfuscate keys and password](#) section of the ASE admin guide. The files containing secured content are:

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

After obfuscating the keys and password, tar and gzip the ASE build and save the build in the `esxi-production/software` directory.

**ABS:** ABS uses the master key defined in `abs_master.key` to obfuscate keys and passwords. Detailed information on generating the master key and securing content is available in the [Obfuscate passwords](#) section of the ABS admin guide. The files containing secured content are:

- `abs.properties` file – `mongo_password`, `jks_password`, and `email_password`
- `abs_init.js` – default password, access, and secret key

After obfuscating the keys and password, tar and gzip the ABS build and save the build in the `esxi-production/software` directory.

**Dashboard:** Dashboard uses the master key defined in `dashboard_master.key` to obfuscate keys and passwords. Detailed information on generating the master key and securing content is available in the [Obfuscating keys and password](#) section of the Dashboard admin guide. The files containing secured content are:

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

After obfuscating the keys and password, tar and gzip the Dashboard build and save the build in the `esxi-production/software` directory.

**Parent topic:** [Step 1: Setup host system](#)

## Change ASE's 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. Complete the following steps to change the default settings:

1. Log in to the management host
2. Change directory to `software`
3. Untar ASE binary:

```
tar -zxvf ase-aws-rhel-3.2.1.tar.gz
```

### 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
```

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]:
```

### 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.

**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 3.2.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

**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 default passwords

Navigate to config directory and edit the following files to change the default key and passwords:

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

### Change CLI 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
```

**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
```

**Next step:** [Import your certificate and repackage ASE](#) for automated deployment.

- [Import existing certificate and repackage ASE](#)

**Parent topic:** [Automated VM setup](#)

## Import existing certificate and repackage ASE

ASE supports both TLS 1.2 and SSLv3 for external APIs.

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-cert.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
```

### Repackage ASE

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

```
tar -zcvf ase-aws-rhel-3.2.1 pingidentity/
```

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

**Parent topic:** [Change ASE's default settings](#)

## Change ABS's 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. Complete the following steps to change the default settings:

1. Log in to the management host

2. Change directory to `software`
3. Untar ABS binary:  

```
tar -zxvf abs-3.2.1.tar.gz
```

### Change default JKS password

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



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

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

```
keytool -storepasswd -keystore config/ssl/abs.jks
Enter keystore password: abs123
New keystore password: newjkspassword
Re-enter new keystore password: newjkspassword
```
2. **Change the key password:** Enter the following command to change the key password. The default key password is `abs123`.  

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

Start ABS after you have changed the default passwords.

### Change `abs_master.key`

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

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

**Stop ABS:** If ABS is running, then stop ABS before generating a new ABS master key. Enter the following command to stop ABS:

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

**Change `abs_master.key`:** Enter the `generate_obfkey` command to change the default ABS master key:

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

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

## Change default passwords

Navigate to `config` directory for `abs.properties` and `mongo` directory `abs_init.js` and edit the following files to change the default key and passwords:

- `abs.properties` file – `mongo_password`, `jks_password`, and `email_password`
- `abs_init.js` – default password, access, and secret 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
```

**Next step:** [Import your certificate and repackage ABS](#) for automated deployment.

- [Import existing certificate and repackage ABS](#)

**Parent topic:** [Automated VM setup](#)

## Import existing certificate and repackage ABS

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==:daYWJ5QgzxZJAnTkuRlFpreM1rsz3F
```

Enter the following command:

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

For example:

```
keytool -importkeystore -destkeystore abs.jks -srckeystore \
abs.p12 -srcstoretype PKCS12 -alias exampleCACertificate

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 3.2.1...
please see /opt/pingidentity/abs/logs/abs/abs.log for more details
```

### Repackage ABS

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

```
tar -zcvf abs-3.2.1 pingidentity/
```

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

**Parent topic:** [Change ABS's default settings](#)

## Change Dashboard default settings

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

1. Log in to the management host
2. Change directory to `software`
3. Untar Dashboard binary:

```
tar -zxvf dashboard-3.2.1.tar.gz
```

### 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-3.2.1 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 host.

**Parent topic:** [Automated VM setup](#)

## 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 has to be run as a `root` user on the Elasticsearch and ASE hosts. The IP address of these hosts was configured in the `hosts` file in [Step 1](#).

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

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

```
[esxi-production]# tail -f logs/ansible.log

=====
Current Time: Mon Feb 11 06:05:25 EST 2019
Starting configure scripts
=====

Mon Feb 11 06:05:25 EST 2019: Setting up local environment
Mon Feb 11 06:05:25 EST 2019: Installing packages
Mon Feb 11 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

Mon Feb 11 06:06:14 EST 2019: Configure successful

=====
configure_esxi_pi.txt
Displaying configure_esxi_pi.txt.
```

## Manually configuring the system parameters

If you cannot run the script as a `root` user, 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:

```
esxi-user soft nofile 65536
esxi-user hard nofile 65536
```

**Parent topic:**[Automated VM setup](#)

### Step 3: Install the PingIntelligence for APIs software

Run the following command to setup the deployment:

```
[esxi-production]# ./bin/start.sh setup
Please see /opt/pingidentity/esxi-production/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:

```
[esxi-production]# tail -f logs/ansible.log
=====
Current Time: Mon Feb 11 06:06:22 EST 2019
Starting setup scripts
=====
Mon Feb 11 06:06:22 EST 2019: Setting up local environment
Mon Feb 11 06:06:22 EST 2019: Installing packages
Mon Feb 11 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

Mon Feb 11 06:23:37 EST 2019: Setup successful
=====
```

**Parent topic:**[Automated VM setup](#)

### Verify Installation

Verify that all the components have installed and started successfully.

#### Verify ASE installation

Log in to the ASE EC2 instance and navigate to `/home/esxi-user/pingidentity/ase/bin` directory and run the `status` command:

```

/home/esxi-user/pingidentity/ase/bin/cli.sh status
Ping Identity Inc., API Security Enforcer
status : started
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>/v3/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": false,
 "api_discovery": {
 "status": false
 },
 "apis": [
 {
 "api_name": "apikeyquery",
 "host_name": "*",
 "url": "/apikeyquery",
 "api_type": "decoy-incontext",
 "creation_date": "Wed Feb 06 19:36:22 IST 2019",
 "servers": 4,
 "protocol": "https",
 "cookie": "",
 "token": false,
 "training_started_at": "training not started yet",
 "training_duration": "24 hours",
 "prediction_mode": false
 },
 {
 "api_name": "apikeyheader",
 "host_name": "*",

```



```

 "url": "/apikeyheader",
 "api_type": "decoy-incontext",
 "creation_date": "Wed Feb 06 19:36:22 IST 2019",
 "servers": 4,
 "protocol": "https",
 "cookie": "",
 "token": false,
 "training_started_at": "training not started yet",
 "training_duration": "24 hours",
 "prediction_mode": false
 },
 {
 "api_name": "atmapp",
 "host_name": "*",
 "url": "/atmapp",
 "api_type": "decoy-incontext",
 "creation_date": "Wed Feb 06 19:36:22 IST 2019",
 "servers": 4,
 "protocol": "https",
 "cookie": "",
 "token": false,
 "training_started_at": "training not started yet",
 "training_duration": "24 hours",
 "prediction_mode": false
 },
 {
 "api_name": "pubatmapp",
 "host_name": "*",
 "url": "/pubatmapp",
 "api_type": "decoy-incontext",
 "creation_date": "Wed Feb 06 19:36:22 IST 2019",
 "servers": 4,
 "protocol": "https",
 "cookie": "JSESSIONID",
 "token": false,
 "training_started_at": "training not started yet",
 "training_duration": "24 hours",
 "prediction_mode": false
 }
],
"abs_cluster": {
 "abs_nodes": [
 {
 "node_ip": "192.168.11.165",
 "os": "Red Hat Enterprise Linux Server release 7.4 (Maipo)",
 "cpu": "24",
 "memory": "62G",
 "filesystem": "76%",
 "bootup_date": "Tue Feb 05 16:12:41 IST 2019"
 }
]
},
"mongodb_nodes": [
 {
 "node_ip": "192.168.11.168",

```

```

 "status": "up"
 }
]
},
"ase_logs": [
 {
 "ase_node": "13eea2fc-64d0-4c51-b663-b1093b0bf7a5",
 "last_connected": "Wed Feb 06 19:41:07 IST 2019",
 "logs": {
 "start_time": "Wed Feb 06 19:36:26 IST 2019",
 "end_time": "Wed Feb 06 19:41:07 IST 2019",
 "gzip_size": "27.51MB"
 }
 }
],
"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"
 }
}
}
}

```

## 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.

**Parent topic:** [Automated VM setup](#)

## Restart PingIntelligence components (optional)

The automated deployment script removes the master key of each PingIntelligence component after the deployment is complete. This is done for security reasons. If you want to restart any PingIntelligence component, the master key using which you have obfuscated the keys and password must be present in the `config` directory of the product.

Complete the following steps to copy the master key back to your PingIntelligence components:

1. Log in to the management host
2. Change directory to `software`
3. Untar the product of which you want to copy the master key
4. Change directory to `config`
5. Copy the master key
6. Log in to host where the specific PingIntelligence component is installed
7. Paste the master key in the `config` directory

**Parent topic:** [Automated VM setup](#)

## Next steps - Integrate PingIntelligence into your environment

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

**Sideband configuration:** If you have 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](#)

**Inline configuration:** If you configured PingIntelligence ASE for [inline connectivity](#), 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](#)
- Adding APIs to ASE based on the deployment mode:
  - [Inline ASE](#), or
  - [Sideband ASE](#)
- [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: 8080
  - 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>:5601/>. 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 default password `changeme`. Change the default password after you log in. The PingIntelligence for APIs Dashboard takes approximately one hour to start showing attack information.

Parent topic:[Automated VM setup](#)

## 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.

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

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

```
[esxi-production]# tail -f logs/ansible.log
=====
Current Time: Mon Feb 11 07:23:11 EST 2019
Starting stop scripts
=====
Mon Feb 11 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
Mon Feb 11 07:32:53 EST 2019: Stop successful
=====
```

Parent topic:[Automated VM setup](#)

## Logs

The `ansible.log` file for all the stages is available in the `/opt/pingidentity/esxi-production/logs` directory.

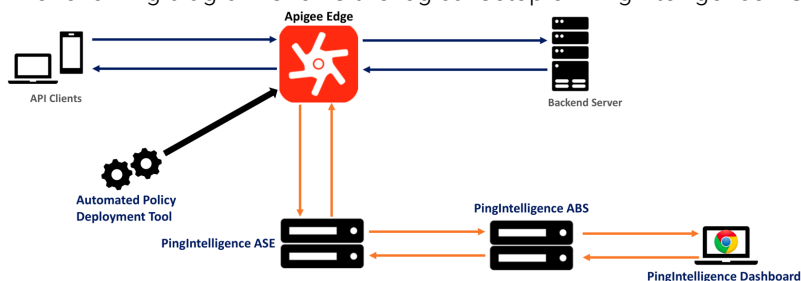
Parent topic:[Automated VM setup](#)

## Apigee Integration

PingIntelligence provides a shared flow to integrate Apigee Edge with PingIntelligence for APIs platform. The two mechanisms of calling shared flows are flow callout and flow hook policies. A Flow Hook in Apigee Edge applies the PingIntelligence shared flow globally to all APIs in an environment under an organization. The Flow Call Out policy in Apigee Edge applies the PingIntelligence shared flow on a per API basis in an environment under an organization.

PingIntelligence provides an automated tool to deploy both Flow Hook and Flow Call Out polices.

The following diagram shows the logical setup of PingIntelligence ASE and Apigee Edge:



Here is the traffic flow through the Apigee Edge and PingIntelligence for APIs components.

1. Incoming request to Apigee Edge
2. Apigee Edge makes an API call to send the request information to ASE
3. ASE checks the request against a registered set of APIs and checks the origin IP, cookie, OAuth2 token or API key against the Blacklist. If all checks pass, ASE returns a 200-OK response to the Apigee Edge. If not, a different response code (403) is sent to Apigee Edge. The request information is also logged by ASE and sent to the AI Engine for processing.
4. If Apigee Edge receives a 200-OK response from ASE, then it forwards the request to the backend server. Otherwise, the Gateway optionally blocks the client.
5. The response from the backend server is received by Apigee Edge.
6. Apigee Edge makes a second API call to pass the response information to ASE which sends the information to the AI engine for processing.
7. ASE receives the response information and sends a 200-OK to Apigee Edge.
8. Apigee Edge sends the response received from the backend server to the client.

## Prerequisites to deploying PingIntelligence shared flow

Confirm that the following prerequisites are met before using the PingIntelligence Apigee tool.

### Prerequisite:

- **Apigee version** - PingIntelligence 3.2.1 works with Apigee Edge Cloud 18.12.04
- Machine where PingIntelligence Apigee tool is installed has JDK 8 installed.
- **PingIntelligence software installation**

PingIntelligence 3.2.1 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-3.2.1.tar.gz
```

- b. To verify that the tool successfully installed, type the `ls` command at the command prompt. This should list the `pingidentity` directory and the build `.tgz` file.

The following table lists the directories:


| Directory        | Description                                                                                                                                                                                                                                                                                                                                                               |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>bin</code> | Contains the following scripts: <ul style="list-style-type: none"> <li>• <code>deploy.sh</code><br/>: The script to deploy the PingIntelligence policy.</li> <li>• <code>undeploy.sh</code><br/>: The script to undeploy the PingIntelligence policy.</li> <li>• <code>status.sh</code><br/>: 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 | <p>Contains the shared flows:</p> <ul style="list-style-type: none"> <li>• request_sharedflow.zip<br/>: Shared flow policy for request</li> <li>• response_sharedflow.zip<br/>: Shared flow for response.</li> </ul> <p>Contains the self-signed certificate that is shipped by default with ASE. The name of the file is ase32.pem</p> |
| 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 | The ASE secondary host IP address and port or hostname and port.<br><div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <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. </div> |
| ase_ssl            | Enable or disable SSL communication between Apigee Edge and ASE. The default value is<br>true<br>.                                                                                                                                                                                                                                                                                  |
| ase_sideband_token | Configure the ASE token generated during the <a href="#">prerequisite</a> step.                                                                                                                                                                                                                                                                                                     |

Following is a sample apigee.properties file:

```
Copyright 2018 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=
#Apigee management server username
apigee_username=
#Apigee management server password
apigee_password=
#Apigee environment to which it should be deployed
apigee_environment=
#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=
```

## 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](#)
- [Deploy PingIntelligence Policy for Flow Call Out](#)
- [Configure PingIntelligence Flow Call Out in Apigee](#)

## 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

**Parent topic:** [Deploy the PingIntelligence policy](#)

## 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
```

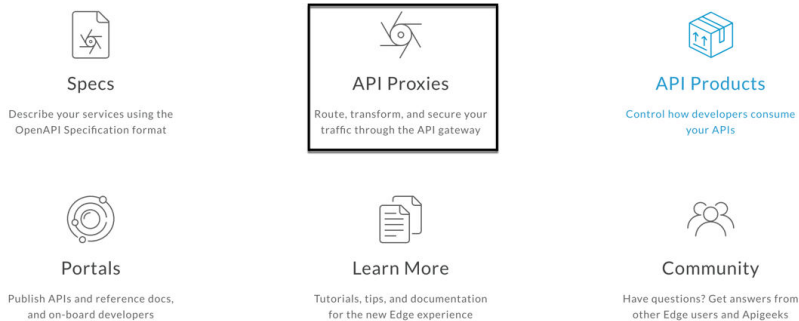
```
PI Apigee Policy is already installed
```

Parent topic: [Deploy the PingIntelligence policy](#)

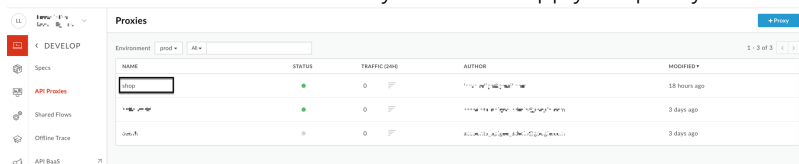
## Configure PingIntelligence Flow Call Out in Apigee

After deploying the Flow Call Out policy using PingIntelligence, configure the PingIntelligence for APIs shared flow. Complete the following steps for Flow Call Out for request and response. The steps listed are for request, complete the same steps for response.

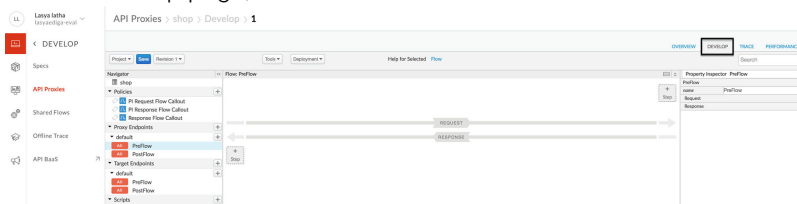
1. Log in to your Apigee Edge account and choose the API Proxy.



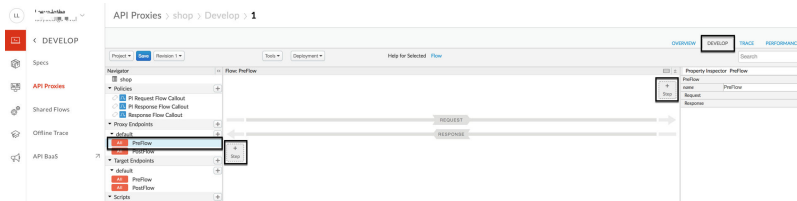
2. Click on the API name on which you want to apply the policy. The Develop page is displayed:



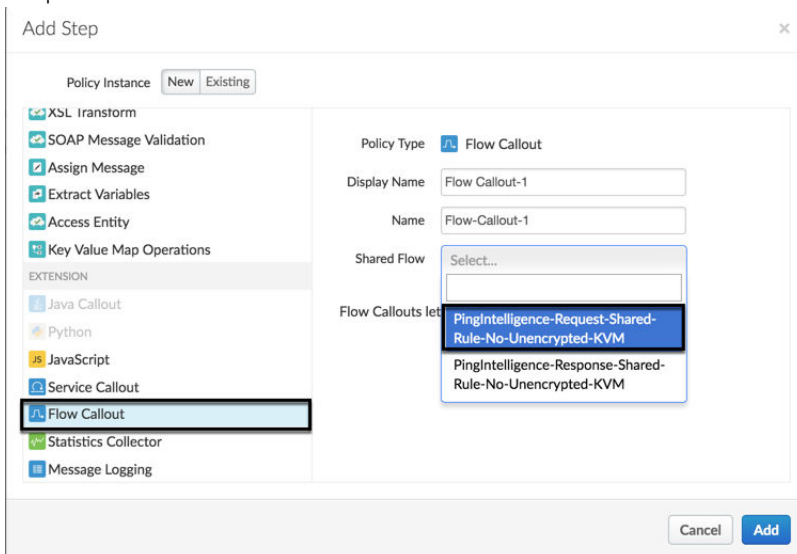
3. On the Develop page, click on the **DEVELOP** tab:



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

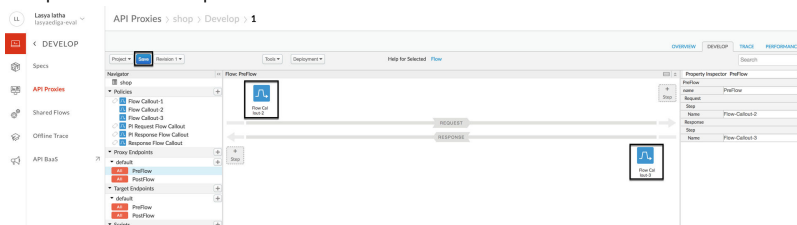


5. In the Add Step window, select **Flow Callout**. From the **Shared Flow** drop down list, select the Request rule and click on **Add**:



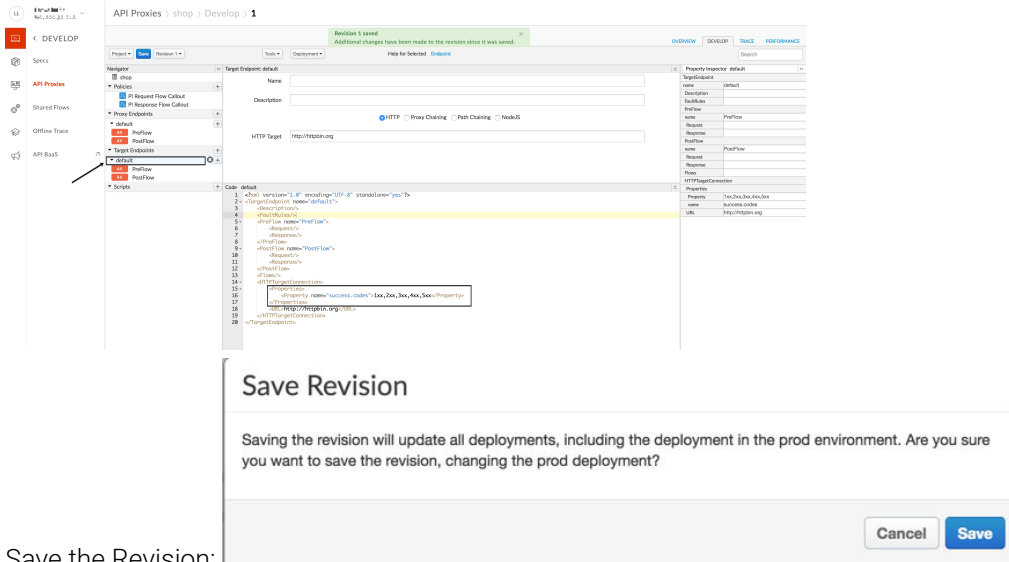
6. Repeat step 5 for Response rule.

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



8. Click on **default** and enter the following lines in the <HTTPTargetConnection> tag:

```
<Properties>
 <Property name="success.codes">1xx,2xx,3xx,4xx,5xx</Property>
</Properties>
```



9. Save the Revision:

Parent topic: [Deploy the PingIntelligence policy](#)

### Change deployed policy mode

You can change the type of policy deployed from Flow Hook to Flow Call Out or Flow Call Out to Flow Hook using the PingIntelligence policy tool. To change the type of policy complete the following steps:

1. Undeploy the deployed policy by entering one of the following command based on the policy and certificate used:

- **Undeploy a Flow Hook policy using self-signed certificate:**

```
/opt/pingidentity/pi/apigee/bin/undeploy.sh -fh
Checking Apigee connectivity
Apigee connectivity ... success
```

Undeploying PI Apigee policy Flow Hook

- 1) Preproxy hook status ... undeployed
- 2) Postproxy hook status ... undeployed
- 3) Request policy undeployment status ... undeployed
- 4) Response policy undeployment status ... undeployed
- 5) Request policy deleting status ... deleted
- 6) Response policy deleting status ... deleted
- 7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
- 8) PingIntelligence-Config-KVM status ... not-applicable
- 9) ASE Primary target server status ... undeployed
- 10) ASE Secondary target server status ... undeployed
- 11) Truststore status ... undeployed
- 12) Cache status ... undeployed

Undeployment of PI Policy finished successfully

- **Undeploy a Flow Hook policy using CA-signed certificate:**

```
opt/pingidentity/pi/apigee/bin/deploy.sh -fh -ca
```

```

Checking Apigee connectivity
Apigee connectivity ... success

Undeploying PI Apigee policy Flow Hook

1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... not-applicable - running using CA signed
certificate
12) Cache status ... undeployed

Undeployment of PI Policy finished successfully

```

- **Undeploy a Flow Call Out policy using self-signed certificate:**

```

/opt/pingidentity/pi/apigee/bin/undeploy.sh -fc
Checking Apigee connectivity
Apigee connectivity ... success

Undeploying PI Apigee policy Flow Call Out

1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... undeployed
12) Cache status ... undeployed

Undeployment of PI Policy finished successfully

```

- **Undeploy a Flow Call Out policy using CA-signed certificate:**

```

opt/pingidentity/pi/apigee/bin/deploy.sh -fc -ca

Checking Apigee connectivity
Apigee connectivity ... success

```

```
Undeploying PI Apigee policy Flow Call Out
```

```
1) Preproxy hook status ... undeployed
2) Postproxy hook status ... undeployed
3) Request policy undeployment status ... undeployed
4) Response policy undeployment status ... undeployed
5) Request policy deleting status ... deleted
6) Response policy deleting status ... deleted
7) PingIntelligence-Encrypted-Config-KVM status ... not-applicable
8) PingIntelligence-Config-KVM status ... not-applicable
9) ASE Primary target server status ... undeployed
10) ASE Secondary target server status ... undeployed
11) Truststore status ... not-applicable - running using CA signed certificate
12) Cache status ... undeployed
```

```
Undeployment of PI Policy finished successfully
```

2. Deploy the other policy by following the steps detailed for [Flow Hook](#) or [Flow Call Out](#)



**Note:** Using the above steps you can also change the use of security certificate from self-signed to CA-signed or from CA-signed to self-signed.

## Add APIs to ASE

After the policy has been deployed to Apigee using the PingIntelligence automated policy tool, add APIs to ASE. Read the following topics to define and add APIs to ASE:

- [API naming guidelines](#)
- [Define and add an API JSON](#)

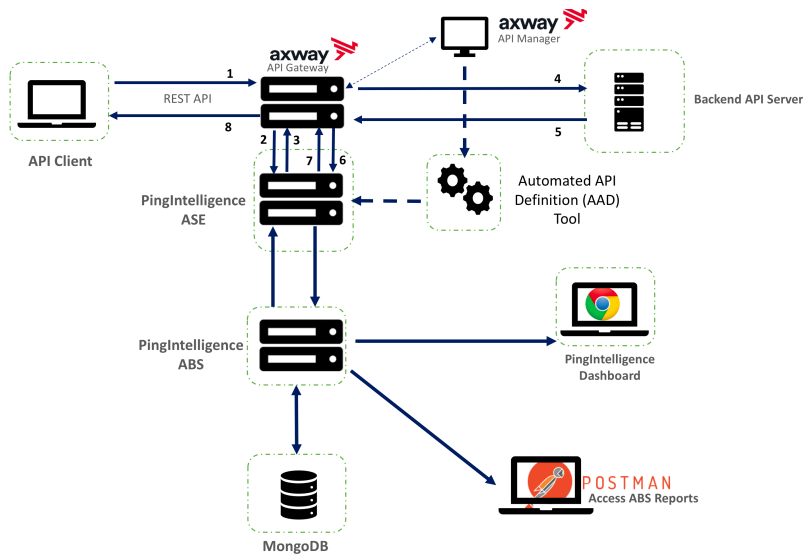
For more information on ASE sideband deployment, see [Sideband API Security Enforcer](#).

## Axway Integration

This guide describes the PingIntelligence - Axway integration steps. For detailed information about ASE in the sideband mode, see the ASE Admin Guide.

The following diagram shows the complete deployment:





Here is the traffic flow through Axway and PingIntelligence for APIs components.

1. Incoming request to Axway
2. Axway 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 or OAuth2 token against the AI 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 for PingIntelligence - Axway Integration](#)
- [Deploy PingIntelligence policy](#)
- [Install and configure PingIntelligence AAD](#)

## Prerequisites - Axway Integration

### Prerequisites:

- **Axway version** PingIntelligence 3.2.1 works with Axway 7.5.3
- **PingIntelligence software installation**

Make sure that PingIntelligence software, ASE, ABS, and Dashboard 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 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 Axway 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 Axway 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.

- **AAD Port** Make sure to open the management port which is used by **AAD** to fetch API definitions from Axway. The default port is 8075. Make sure to 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.

**Parent topic:** [PingIntelligence Axway API Gateway Integration](#)

## Deploy PingIntelligence policy

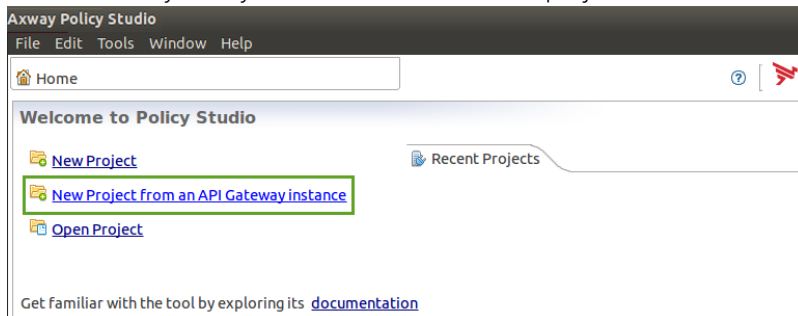
Deploying PingIntelligence policy requires completing the following two parts:

- *Configuring Axway Policy Studio*
- *Configuring Axway API Manager*
- [Axway Policy Studio configuration](#)
- [Axway API Manager configuration](#)

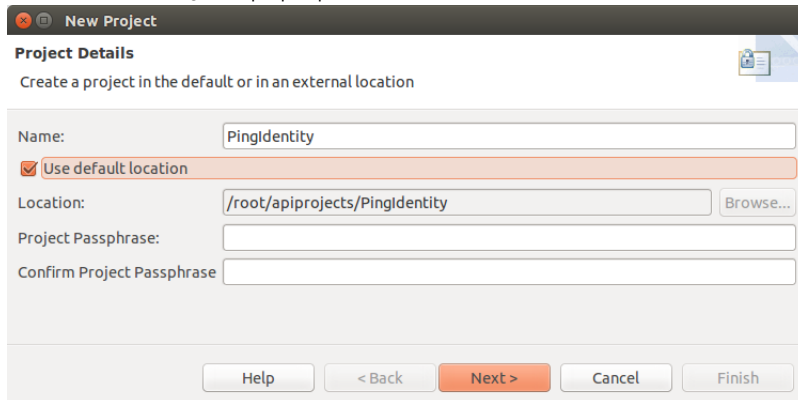
**Parent topic:** [PingIntelligence Axway API Gateway Integration](#)

## Axway Policy Studio configuration

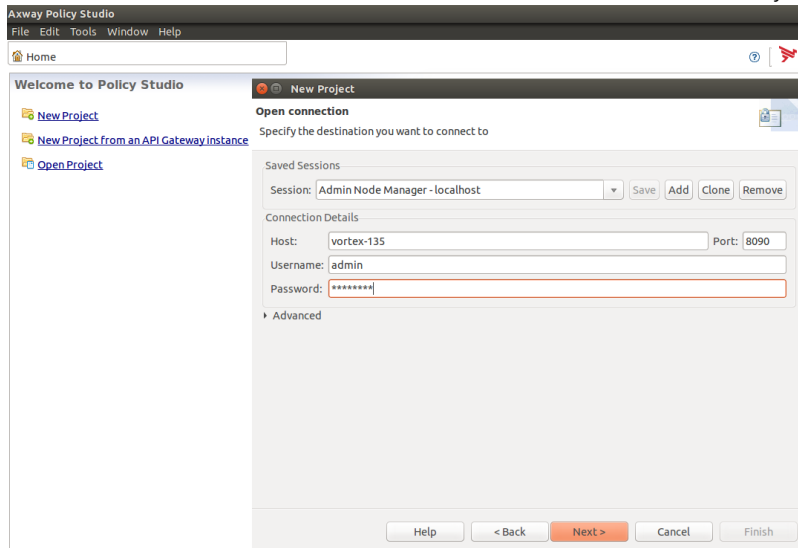
1. Launch Axway Policy Studio and create a new project from an **API Gateway instance**:



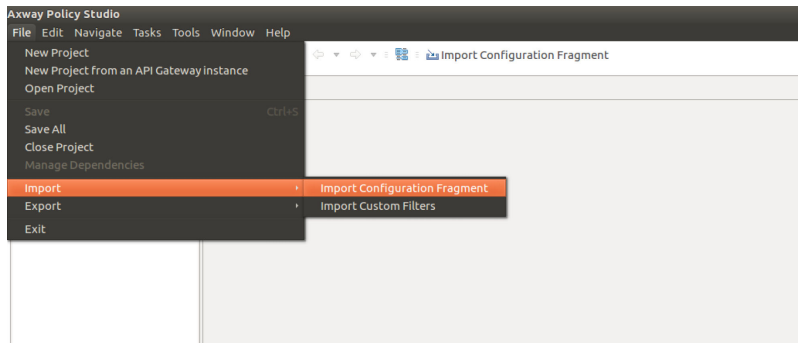
2. In the **New Project** pop-up window, enter the details and click **Next**:



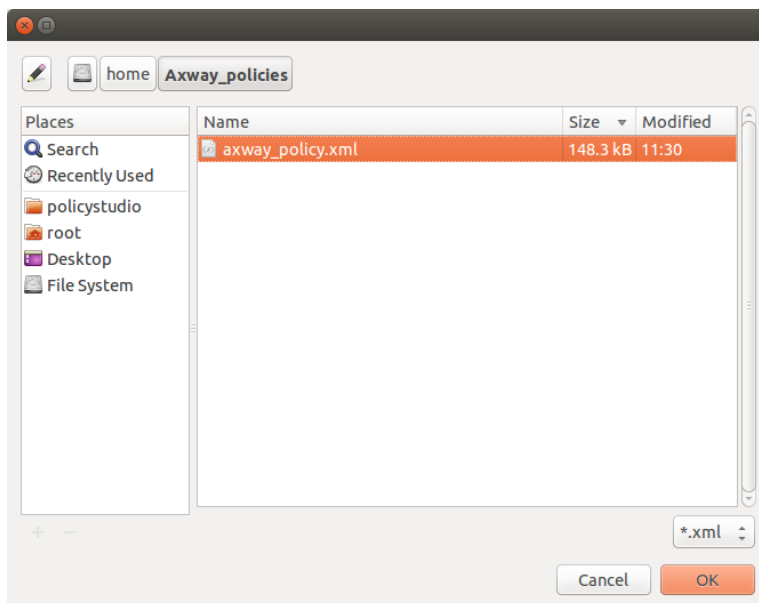
3. Enter **Host** details, **Username**, and **Password** of the API Gateway to connect and click **Next**:



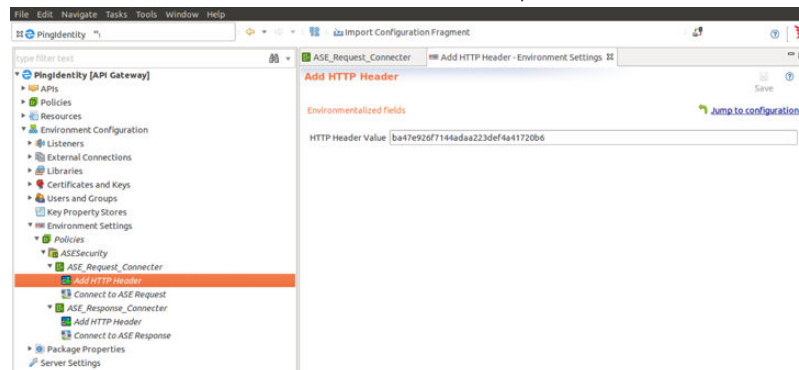
4. Click **Import configuration fragment** from the **File** sub menu in the menu bar



From the pop-up window, import the Axway Policy from the directory where it was saved. Select the policy and click **OK**:

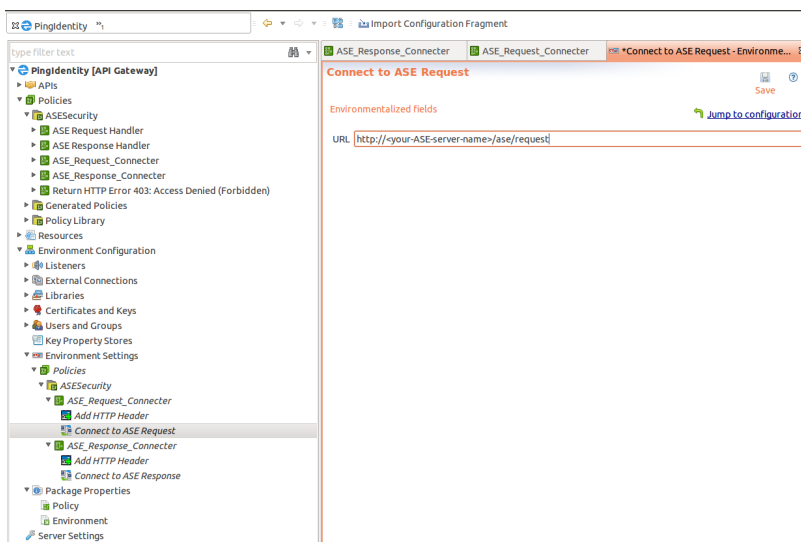


5. After the Axway Policy is imported, click **Environment Settings** in the left-hand column and Click **Add HTTP Header**. In the HTTP Header Value field, enter the ASE authentication token that was

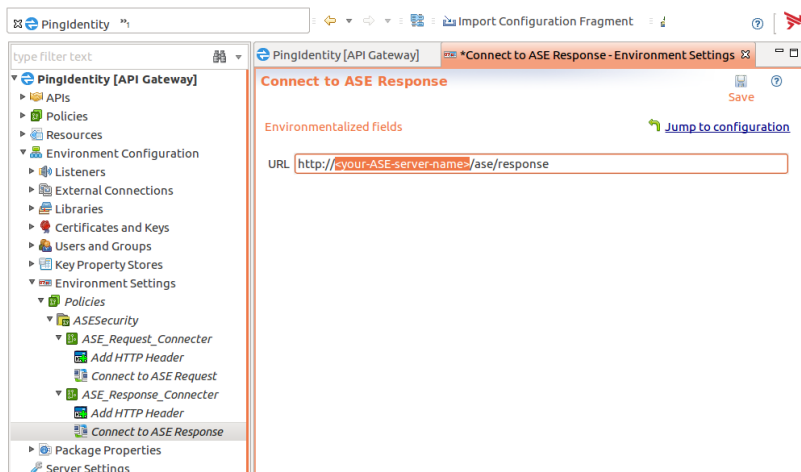


created.

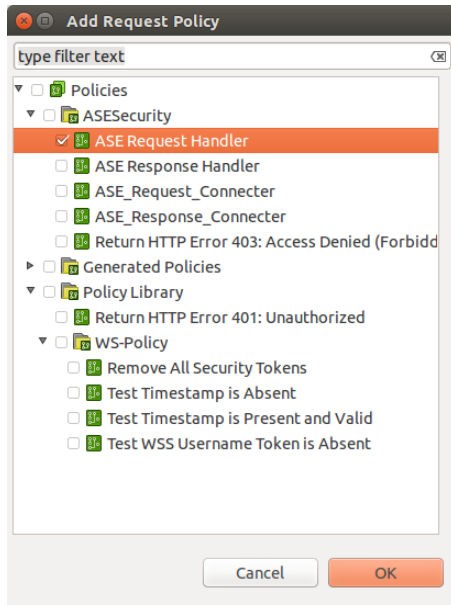
6. After the Axway Policy is imported, click **Environment Settings** in the left-hand column and click **Connect to ASE Request** under **ASE\_Request\_Connector**. Enter the IP address or the hostname of your ASE in the **URL** field as shown in the screen shot:



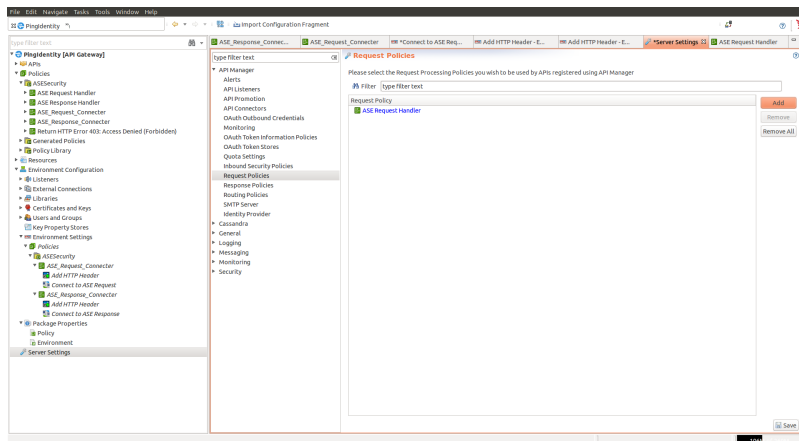
7. In the **Environment Settings** in the left-hand column, click **Connect to ASE Response** under **ASE\_Response\_Connector**. Enter the IP address or the hostname of your ASE in the **URL** field as shown in the screen shot:



8. In the left pane of the window, click **Server Settings**.
9. In the Server Settings window, double-click **Request Policies** under **API Manager**
10. In the **Add Request Policy** pop-up window, check the **ASE Request Handler** and click **OK**



11. Click **Add** and then **Save**



12. Repeat step 9-10 for Response Policies.
13. Deploy the Policies by clicking **Deploy**.

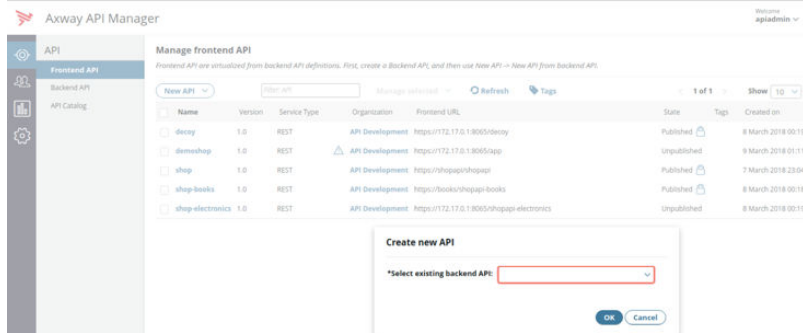
Parent topic: [Deploy PingIntelligence policy](#)

## Axway API Manager configuration

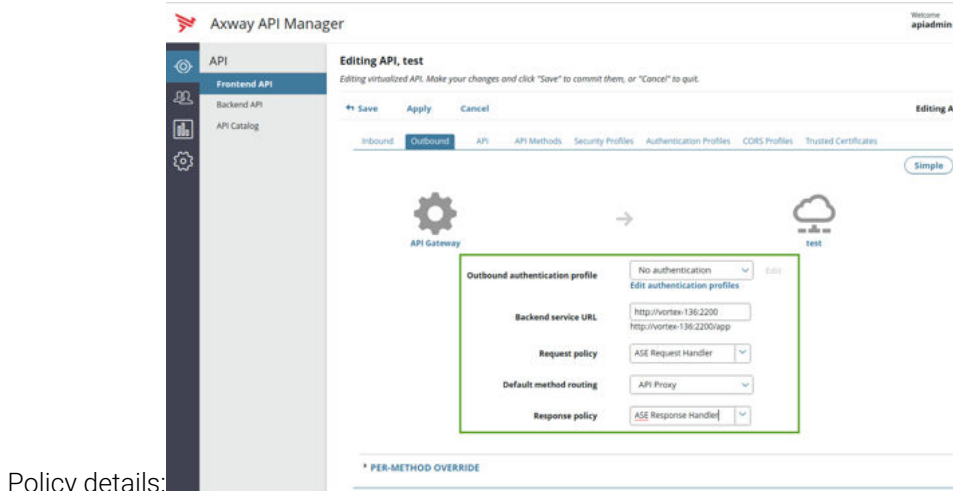
Complete the following steps to configure Axway API Manager:

1. Login to the Axway API Manager.

2. In the Axway API Manager, click **Frontend API** and **Create new API**



3. Click **Outbound tab** and enter Backend Service URL (your backend application server) and Request



Policy details:

Parent topic: [Deploy PingIntelligence policy](#)

## Install and configure AAD

Install and configure AAD to capture API definitions from your Axway API gateway. These APIs are discovered by AAD and converted into API JSON by AAD and added to ASE.

### Install AAD

Download the AAD tool from the [download](#) site. Oracle Java 8 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-3.2.1.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.mode</code>	Set the value to <code>gateway</code> when ASE is deployed in sideband mode. For more information on ASE modes, see the <i>ASE Admin Guide</i> .
<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>	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>	URL of the API Gateway. Only valid when <code>aad.mode</code> is <code>gateway</code> .
<code>gateway.management.username</code>	Username to connect to the API Gateway. Only valid when <code>aad.mode</code> is <code>gateway</code> .



gateway.management.password	Password to connect to the API Gateway. Only valid when aad.mode is gateway .
-----------------------------	----------------------------------------------------------------------------------------------

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:Rs7NPeYGCU0Zku7TANJbwE12rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwE12rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
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==:0vOu0XUVpbvV4AaSmv5mZ1lw3WpAsj1oPF3d5Et170Y=
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: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/UN3Efz0DMa/
kmI=:Az82rlUFftMGpMxF7unelJZUucX1911O2QgKvHD36vU=
```

## 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](#)
- [OAuth2 Token and API Keys](#)

**Parent topic:** [PingIntelligence Axway API Gateway Integration](#)

## 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](#)
- [Configure tags for decoy API](#)

Parent topic: [Install and configure PingIntelligence AAD](#)

## 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 - true
```

### 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:

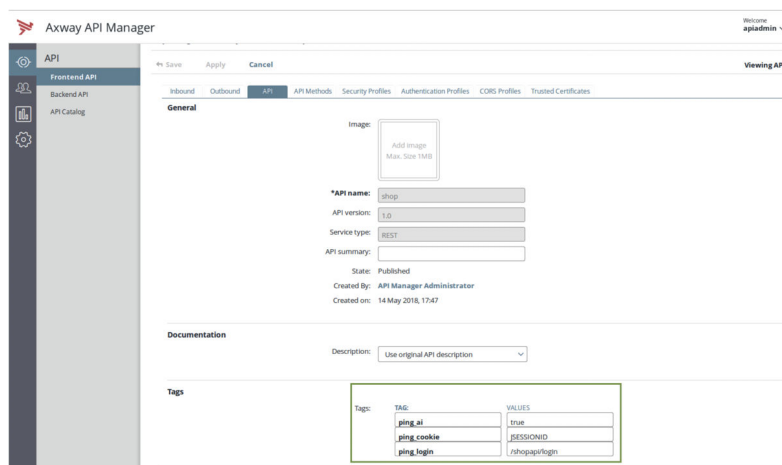
- `ping_cookie - JSESSIONID`

- `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:



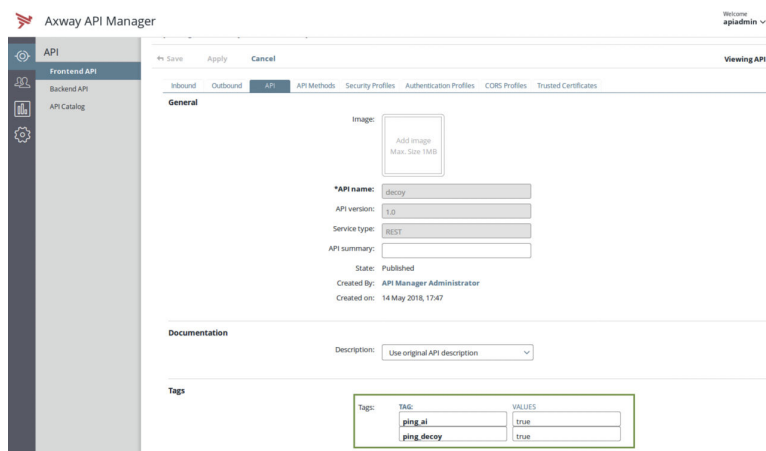
Parent topic: [Axway API Manager configuration for AAD](#)

## Configure tags for decoy API

You can configure Decoy APIs in Axway API Manager. A Decoy API is an API for which the traffic does not reach the backend API servers. The Decoy API is deployed to gather information about potential threats that your API ecosystem may face. Traffic directed to Decoy API configured in Axway API Gateway is redirected to ASE which functions as the backend server. ASE sends a preconfigured response, like 200 OK, for requests sent to a Decoy API.

You need to configure the following **TAGS** and **VALUES** in the **API** tab for **Frontend API** in Axway API Manager:

- ping\_ai - true
- ping\_decoy - true



**API JSON for decoy API:** The converted API JSON will have the decoy section configured as highlighted in the following JSON file:

```
{
 "api_metadata": {
 "protocol": "https",
 "url": "/decoy",
 "hostname": "*",
 "cookie": "",
 "cookie_idle_timeout": "",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": false,
 "apikey_qs": "",
 "apikey_header": "",
 "enable_blocking": true,
 "login_url": "",
 "api_mapping": {
 "internal_url": ""
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
```

```

 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "",
 "error_def": "",
 "error_message_body": ""
},
"flow_control": {
 "client_spike_threshold": "0/second",
 "server_connection_queueing": false
},
"api_memory_size": "64mb",
"health_check": false,
"health_check_interval": 60,
"health_retry_count": 4,
"health_url": "/",
"server_ssl": false
"servers": [],
"decoy_config": {
 "decoy_enabled": true,
 "response_code": 200,
 "response_def": "OK",
 "response_message": "OK",
 "decoy_subpaths": []
}
}
}

```

Parent topic: [Axway API Manager configuration for AAD](#)

## OAuth2 Token and API Keys

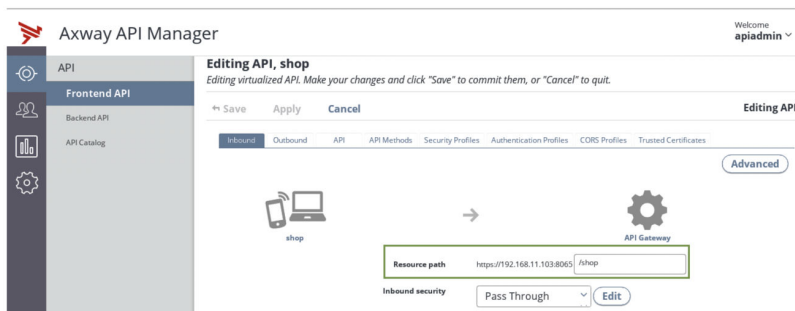
If you have configured the API Key in Request Header or in Query String, AAD reads and converts these values to `apikey_qs` or `apikey_header` values in the ASE API JSON. PingIntelligence's AI engine considers API Key values only in request headers or the query string.

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.



```
{
 "api_metadata": {
 "protocol": "https",
 "url": "/shop",
 "hostname": "192.168.11.103",
 "cookie": "JSESSIONID",
 "cookie_idle_timeout": "",
 "logout_api_enabled": false,
 "cookie_persistence_enabled": false,
 "oauth2_access_token": true,
 "apikey_qs": "KeyId",
 "apikey_header": "",
 "enable_blocking": true,
 "login_url": "/shop/login",
 "api_mapping": {
 "internal_url": ""
 },
 },
 "api_pattern_enforcement": {
 "protocol_allowed": "",
 "http_redirect": {
 "response_code": "",
 "response_def": "",
 "https_url": ""
 },
 "methods_allowed": [],
 "content_type_allowed": "",
 "error_code": "",
 "error_def": "",
 "error_message_body": ""
 },
 "flow_control": {
 "client_spike_threshold": "0/second",
 "server_connection_queueing": false
 },
 "api_memory_size": "64mb",
 "health_check": false,
 "health_check_interval": 60,
 "health_retry_count": 4,
 "health_url": "/",
 "server_ssl": false,
 "servers": [],
 "decoy_config": {
 "decoy_enabled": false,
 }
}
```

```

 "response_code": 200,
 "response_def": "",
 "response_message": "",
 "decoy_subpaths": []
 }
}
}

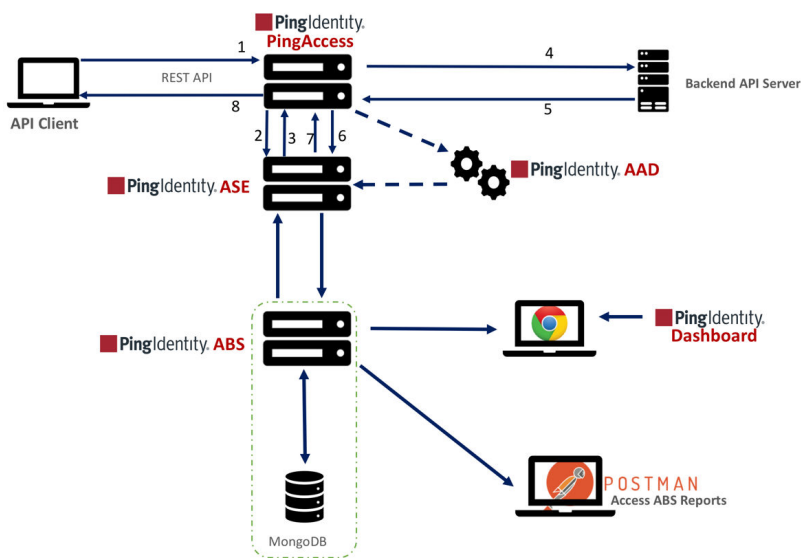
```

Parent topic: [Install and configure PingIntelligence AAD](#)

## PingAccess Integration

This guide describes the installation of PingIntelligence for APIs 3.2.1 with PingAccess 5.2.

This diagram depicts the architecture of PingIntelligence for APIs components along with PingAccess:



Here is the traffic flow through the PingAccess and PingIntelligence for APIs components.

1. Incoming request to PingAccess
2. PingAccess 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 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.
4. 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.
5. The response from the backend server is received by PingAccess.
6. PingAccess 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 PingAccess.
8. PingAccess sends the response received from the backend server to the client.

- [Prerequisites for PingIntelligence - PingAccess Integration](#)
- [Deploy the PingIntelligence policy](#)
- [Install and configure PingIntelligence AAD](#)



## Prerequisites - PingAccess Integration

Make sure that the following prerequisites are met before configuring PingAccess:

- **PingAccess version** - PingIntelligence 3.2.1 works with PingAccess 5.2.
- **PingIntelligence software installation**

Make sure that PingIntelligence software, ASE, ABS, and Dashboard 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 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 PingAccess 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 PingAccess 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.

- **Port for AAD**

Make sure to open the management port which is used by [AAD](#) to fetch API definitions from PingAccess. The default port is 9000. Make sure to open port 8010 in ASE for AAD to add API definitions.

**Parent topic:** [PingIntelligence – PingAccess Integration](#)

## Deploy the PingIntelligence policy

To integrate PingAccess with PingIntelligence components, complete the following steps on PingAccess:

1. Download the Ping Access Policy from the [download](#) site and unzip it.
2. Copy the `PingIntelligence.jar` file in the `lib` directory in `PA_home`.
3. Restart PingAccess
4. Add Applications in PingAccess with **Application Type** as API. Make sure that the following description is added in the **DESCRIPTION** section when you add an Application:

```
{
 "ping_ai": true,
 "ping_host": "",
 "ping_url": "",
 "ping_login": "",
 "ping_cookie": "",
 "apikey_qs": "",
 "apikey_header": "",
 "ping_decoy": false,
 "oauth2_access_token": false
}
```

The following table describes the parameters. This description is required for [AAD](#) to fetch the API definition from PingAccess and add to ASE:

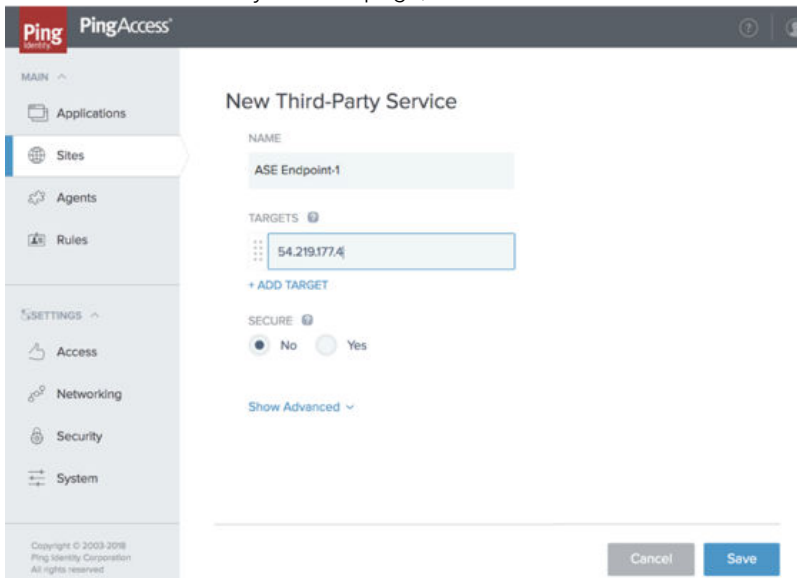
Parameter	Description
<code>ping_ai</code>	Configures whether the artificial intelligence (AI) processing should be carried out on this API or not. Default value is <code>true</code> . Set to <code>false</code> if you do not want AI processing on an API.
<code>ping_host</code>	Hostname of the API. You can also configure <code>*</code> , <code>as</code> , <code>hostname</code> , or <code>.</code>
<code>ping_url</code>	The URL of the managed API, for example, <code>/shopping</code> . This field cannot be empty.
<code>ping_login</code>	Login URL for the API. The field can be empty.
<code>ping_cookie</code>	Cookie name for the API. The field can be empty.
<code>apikey_qs</code>	When API Key is sent in the query string, ASE uses the specified parameter name to capture the API key value.
<code>apikey_header</code>	When API Key is part of the header field, ASE uses the specified parameter name to capture the API key value.

ping_decoy	Whether API is a decoy or not. The values can be true or false.
oauth2_access_token	Whether API is OAuth2 aware or not. The values can be true or false.

5. Add two ASEs to Third-Party Services.

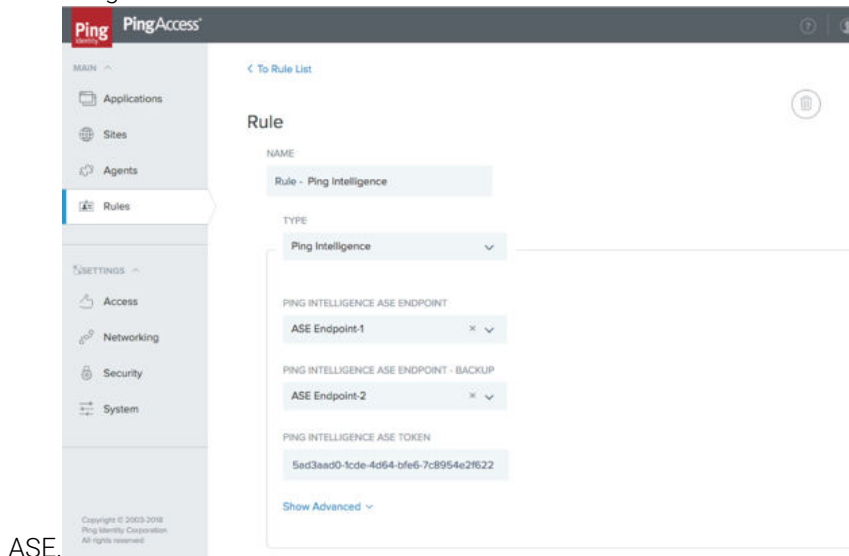


In the New Third-Party Service page, add ASE IP address. Add two separate sites for ASE.



6. 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.
7. Select the **ASE SERVICE ENDPOINT** to which you want to apply the rule.

8. Add the generated ASE sideband token which is used for authentication between PingAccess and



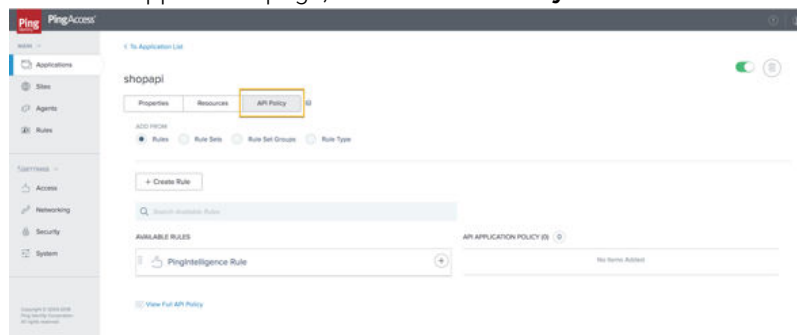
ASE.





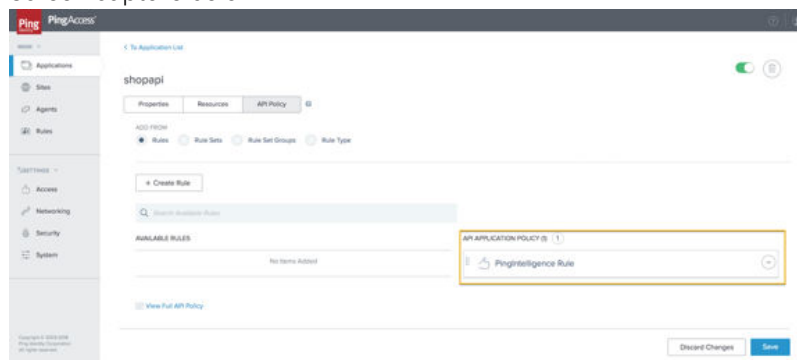
**Note:** In case of an ASE node failure, traffic is automatically routed to the standby ASE node.

9. 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  sign for PingIntelligence rule. After clicking on the  sign, the PingIntelligence Rule moves under 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.

**Parent topic:** [PingIntelligence – PingAccess Integration](#)

## Install and configure AAD

Install and configure AAD to capture API definitions from your Axway API gateway. These APIs are discovered by AAD and converted into API JSON by AAD and added to ASE. For detailed information on AAD, see Automated API Definition tool information in the [ABS admin guide](#).

### Install AAD

Download the AAD tool from the [download](#) site. Oracle Java 8 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-3.2.1.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.mode</code>	Set the value to <code>pingaccess</code> when ASE is deployed in sideband mode. For more information on ASE modes, see the <i>ASE Admin Guide</i> .
<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>	Set to true for ABS-AAD communication to use SSL.
<code>ase.access_key</code>	The username of ASE. Default value is

	admin
ase.secret_key	The password of ASE. Default value is admin
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
pingaccess.management.url	URL of the API Gateway. Only valid when aad.mode is pingaccess .
pingaccess.management.username	Username to connect to the API Gateway. Only valid when aad.mode is pingaccess .
pingaccess.management.password	Password to connect to the API Gateway. Only valid when aad.mode is pingaccess .

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=pingaccess
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:Rs7NPeYGCU0Zku7TANJbwE12rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
```

```

ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGCU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
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: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/UN3E fz0DMA/
kmI=:Az82rlUFftMGpmxF7une1JZUucX1911O2QgKvHD36vU=

```

## 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.
```

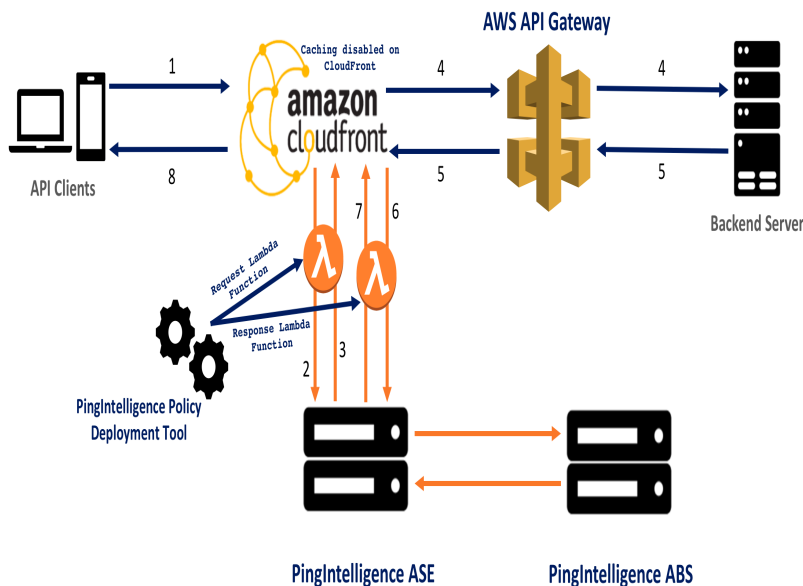
Parent topic: [PingIntelligence – PingAccess Integration](#)

## AWS API Gateway Integration

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 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 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 to deploying the PingIntelligence Lambda Policy

The following prerequisites must be met before running the PingIntelligence AWS policy tool.

### Prerequisite:

- JDK 8 must be installed on the system running the PingIntelligence policy tool.
- **PingIntelligence software installation**

PingIntelligence 3.2.1 software are installed and configured. For installation of PingIntelligence software, see the manual or platform specific automated deployment guides.

- **AWS admin account:** To deploy PingIntelligence policy, you must have AWS admin account.
- **Edit CloudFront configuration:** Make sure that 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:** Make sure that the **Minimum TTL**, **Maximum TTL**, and the **Default TTL** is 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, DELETE ⓘ

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 ⓘ

Compress Objects Automatically  Yes  No ⓘ [Learn More](#)

- **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.

## 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-aws-3.2.1.tar.gz
```


- b. To verify that the tool successfully installed, type the `ls` command at the command prompt. This should list the `pingidentity` directory and the `build .tgz` file.


The following table lists the directories:

Directory	Description
bin	Contains the following scripts: <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> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <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>role</code> mode.</p> </div>
access_key	AWS access key. This is applicable when the mode is set to

	keys
secret_key	AWS secret key. This is applicable when the mode is set to keys
aws_lambda_memory	AWS Origin Response Lambda memory in MB. Default value is 1024 MB. The memory can be configured in multiple of 64. Minimum and maximum value are 128 and 3008 respectively. For more information, see <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> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;">  <b>Note:</b> This field cannot be left blank. In a testing environment, enter the same IP address for primary and secondary ASE host. </div> <p>If both the ASE hosts are unreachable, the request is directly sent to the backend API server.</p>
ase_ssl	Enable or disable SSL communication between Lambda functions and ASE. The default value is true.
ase_sideband_token	Enter the ASE token generated during the <a href="#">prerequisite</a> step.

Following is a sample `aws.properties` file:

```
Copyright 2019 Ping Identity Corporation. All Rights Reserved.
Ping Identity reserves all rights in The program as delivered.
Unauthorized use, copying,
modification, reverse engineering, disassembling, attempt to discover any
source code or
underlying ideas or algorithms, creating other works from it, and
distribution of this
program is strictly prohibited. The program or any portion thereof may not
be used or
reproduced in any form whatsoever except as provided by a license without
the written
```

```
consent of Ping Identity. A license under Ping Identity's rights in the
Program may be
available directly from Ping Identity.

#Authentication mode access-key & secret-key / role based access. Values can
be keys or role.
mode=keys
#AWS access key
access_key=AKIAID7MDWSCUUVHMTNA
#AWS secret key
secret_key=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
#AWS Lambda memory in MB. It should be a multiple of 64. Minimum and maximum
value are 128 and 3008 respectively.
aws_lambda_memory=1024
#Cloudfront distribution ID
cloudfront_distribution_id=EGQ9OEG3ZDABP

#ASE Primary Host <IP/Host>:<port>
ase_host_primary=test.elasticbeam.com
#ASE Secondary Host <IP/Host>:<port>
ase_host_secondary=test.elasticbeam.com
#ASE SSL status
ase_ssl=true
#ASE sideband authentication token
ase_sideband_token=283ded57cd5f48e6bcd8fa3ba9d2888d
```

## Create Role

If you have set the authentication mode as role in the `aws.properties` file, create a role for the EC2 instance. This role is required for the PingIntelligence policy deployment tool. Complete the following steps to create and configure.

1. Select EC2 as service and click on **Next: Permissions** button:

The screenshot shows the AWS IAM console 'Create role' wizard. The first step is 'Select type of trusted entity', with 'AWS service' selected. The second step is 'Choose the service that will use this role', where 'EC2' is selected. A grid of services is displayed, with 'EC2' highlighted. The 'Next: Permissions' button is visible at the bottom right.

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		

2. Choose the following three Policies and provide a name for each role (for example, PIDeploymentToolRole):

- AWSLambdaFullAccess
- CloudFrontFullAccess
- AmazonEC2FullAccess

After providing the name, click on **Create role**.

The screenshot shows the 'Create role' page in the AWS IAM console, specifically the 'Review' step. The page has a dark header with the AWS logo and navigation options. Below the header, there are four numbered steps (1, 2, 3, 4) with step 4 being the active one. The main content area is titled 'Review' and contains the following information:

- Role name\***: PIDeploymentToolRole. Below it, a note says 'Use alphanumeric and '+', '@', '-' characters. Maximum 64 characters.'
- Role description**: Allows EC2 instances to call AWS services on your behalf. Below it, a note says 'Maximum 1000 characters. Use alphanumeric and '+', '@', '-' characters.'
- Trusted entities**: AWS service: ec2.amazonaws.com
- Policies**: Three policies are listed: AWSLambdaFullAccess, CloudFrontFullAccess, and AmazonEC2FullAccess, each with a link icon.
- Permissions boundary**: Permissions boundary is not set.

At the bottom, it says 'No tags were added.' and there are three buttons: 'Cancel', 'Previous', and 'Create role'.

3. In the Summary page of the role that you created in step 2, click on the **Trust relationships** tab and then click on **Edit trust relationship** button:

The screenshot shows the 'Summary' page for the role 'PIDeploymentToolRole' in the AWS IAM console. The page has a dark header with the AWS logo and navigation options. On the left, there is a sidebar with navigation options: Dashboard, Groups, Users, Roles (selected), Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area is titled 'Summary' and contains the following information:

- Role ARN**: arn:aws:iam::377367197819:role/PIDeploymentToolRole
- Role description**: Allows EC2 instances to call AWS services on your behalf. | Edit
- Instance Profile ARNs**: arn:aws:iam::377367197819:instance-profile/PIDeploymentToolRole
- Path**: /
- Creation time**: 2019-02-07 14:11 UTC+0530
- Maximum CLI/API session duration**: 1 hour | Edit

Below this information, there are five tabs: 'Permissions', 'Trust relationships' (selected), 'Tags', 'Access Advisor', and 'Revoke sessions'. Under the 'Trust relationships' tab, there is a button 'Edit trust relationship'. Below this button, there are two sections:

- Trusted entities**: The following trusted entities can assume this role. There are two entities listed: 'The identity provider(s) ec2.amazonaws.com' and 'The identity provider(s) lambda.amazonaws.com'.
- Conditions**: The following conditions define how and when trusted entities can assume the role. There are no conditions associated with this role.

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"
}
]
}

```

aws Services Resource Groups

Edit Trust Relationship

You can customize trust relationships by editing the following access control policy document.

Policy Document

```

1 {
2 "Version": "2012-10-17",
3 "Statement": [
4 {
5 "Effect": "Allow",
6 "Principal": {
7 "Service": "ec2.amazonaws.com"
8 },
9 "Action": "sts:AssumeRole"
10 },
11 {
12 "Effect": "Allow",
13 "Principal": {
14 "Service": "lambda.amazonaws.com"
15 },
16 "Action": "sts:AssumeRole"
17 }
18]
19 }

```

Cancel Update Trust Policy

## 5. Configure the **IAM role**, as the role that you created (for example, PIDeploymentToolRole):

aws Services Resource Groups PingIntelligence © 2022/1/23/23 N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances 1 Launch into Auto Scaling Group

Purchasing option  Request Spot instances

Network vpc-6e11152b (Default (default)) Create new VPC

Subnet No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP Use subnet setting (Enable)

Placement group  Add instance to placement group

Capacity Reservation Open Create new Capacity Reservation

IAM role PIDeploymentToolRole Create new IAM role

Shutdown behavior Stop

Enable termination protection  Protect against accidental termination

Monitoring  Enable CloudWatch detailed monitoring Additional charges apply

Tenancy Shared - Run in shared hardware instance Additional charges will apply for dedicated tenancy

Elastic Inference  Add an Elastic Inference accelerator Additional charges apply

T3/T3 Unlimited  Enable Additional charges may apply

Advanced Details

Cancel Previous Review and Launch Next: Add Storage



## Deploy PingIntelligence Policy for AWS

Using the PingIntelligence AWS policy tool, deploy the PingIntelligence policy in AWS @Lambda in North Virginia (US-East-1) region. Note that the policy must currently be deployed in this region. The Lambda function pushes the PingIntelligence policy to Amazon CloudFront in the local AWS instances. The PingIntelligence Lambda policy communicates with PingIntelligence ASE to collect request and response data 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. The self-signed certificate is part of 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 PingIntelligence into your 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 you have added your APIs in ASE, the API model needs to be trained. The training of the API model is completed in 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.**

## Undeploy PingIntelligence AWS Policy

Undeploy the PingIntelligence AWS policy using the undeploy tool which detaches the policy from CloudFront. The time to detach the policy from CloudFront depends on which CloudFront region the policy is deployed.

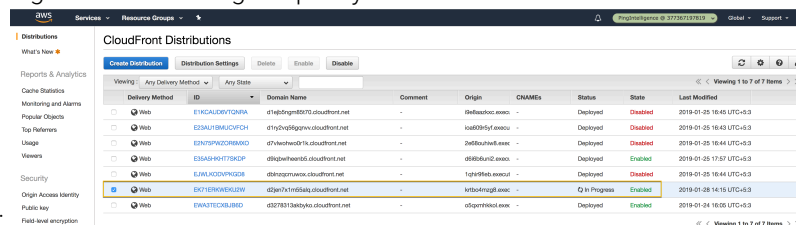
To undeploy the policy, run the following command:

```
/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.

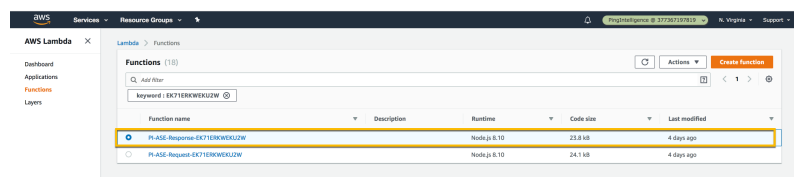
Check the progress of detaching the policy from the CloudFront in the AWS GUI as shown in the following



Delivery Method	ID	Domain Name	Comment	Origin	CHNAMEs	Status	State	Last Modified
Web	E1KCALQK7QYQNA	d1e6p8ng-d8570.cloudfront.net	-	filebackloc.exeec	-	Deployed	Disabled	2019-01-25 16:45 UTC+0:0
Web	E58AUI8WJQVQGH	d17y2v2p2lqgnv.cloudfront.net	-	load6981yf.exeec	-	Deployed	Disabled	2019-01-25 16:43 UTC+0:0
Web	E2NTSPWZQW8WGD	d17v2v2p2lqgnv.cloudfront.net	-	zef5d9u7w.exeec	-	Deployed	Disabled	2019-01-25 16:44 UTC+0:0
Web	E58A8KH778KDP	d18q2w7e4e05.cloudfront.net	-	d1866AUS.exeec	-	Deployed	Enabled	2019-01-25 17:57 UTC+0:0
Web	E1WAKQZVW9KGB	d18q2w7e4e05.cloudfront.net	-	1q8d9f8b.exeec	-	Deployed	Disabled	2019-01-25 16:44 UTC+0:0
Web	EK71ERW6KZUW	d2j617x1m66u.cloudfront.net	-	ltdb07r0qg.exeec	-	In Progress	Enabled	2019-01-28 14:15 UTC+0:0
Web	EWA7TECKLJBG	d3278313akyo.cloudfront.net	-	d5qpm7k0ed.exeec	-	Deployed	Enabled	2019-01-24 16:00 UTC+0:0

screenshot:

After the **State** has moved from Enabled to Disabled, delete the Request and Response Lambda functions. Use the `cloud_front_id` from the `aws.properties` file to search for PingIntelligence Lambda functions.



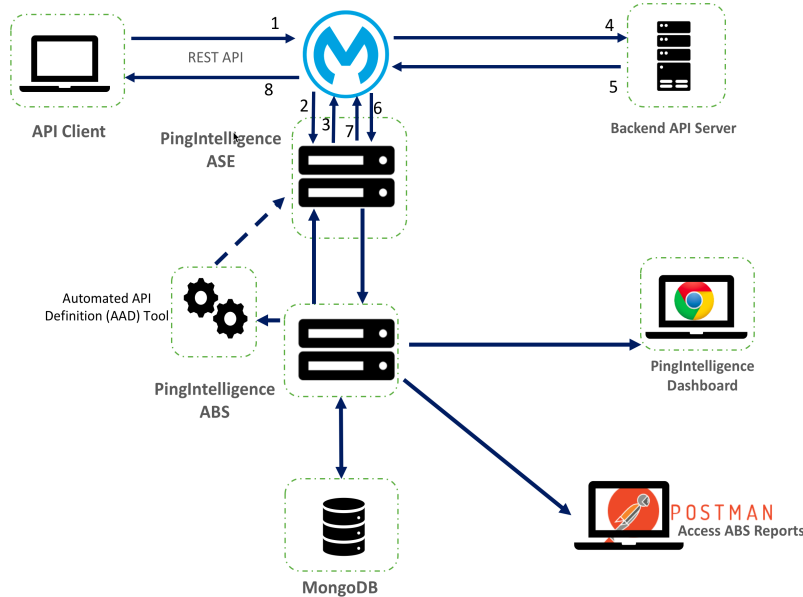
Function name	Description	Runtime	Code size	Last modified
PI-ASE-Response-EK71ERW6KZUW		Node.js 8.10	23.8 kB	4 days ago
PI-ASE-Request-EK71ERW6KZUW		Node.js 8.10	24.1 kB	4 days ago



**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.

## Mulesoft API Gateway Integration

PingIntelligence provides a policy to integrate PingIntelligence ASE 3.2.1 and Mulesoft Anypoint API gateway. The following diagram shows the logical setup of PingIntelligence ASE and Mulesoft:



Here is the traffic flow through the Mulesoft and PingIntelligence for APIs components.

1. Incoming request to Mulesoft
2. Mulesoft 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 Mulesoft. If not, a different response code (403) is sent to Mulesoft. The request information is also logged by ASE and sent to the AI Engine for processing.
4. If Mulesoft receives a 200-OK response from ASE, then it forwards the request to the backend server. Otherwise, the Mulesoft optionally blocks the client.
5. The response from the backend server is received by Mulesoft.
6. Mulesoft 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 Mulesoft.
8. Mulesoft sends the response received from the backend server to the client.

## Prerequisites to deploying PingIntelligence Policy

Confirm that the following prerequisites are met before deploying PingIntelligence policy:

### Prerequisite:

- **PingIntelligence software installation**

PingIntelligence 3.2.1 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 Mulesoft Anypoint 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 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.

## Deploy PingIntelligence policy

PingIntelligence provides a policy to deploy PingIntelligence 3.2.1 with Mulesoft 3.9 and 4.0. The policy package has the following two files, an `xml` and a `yaml`:

- `pi_policy.yaml`
- `pi_policy.xml`

Follow the steps to deploy PingIntelligence policy based on the version of Mulesoft API gateway.

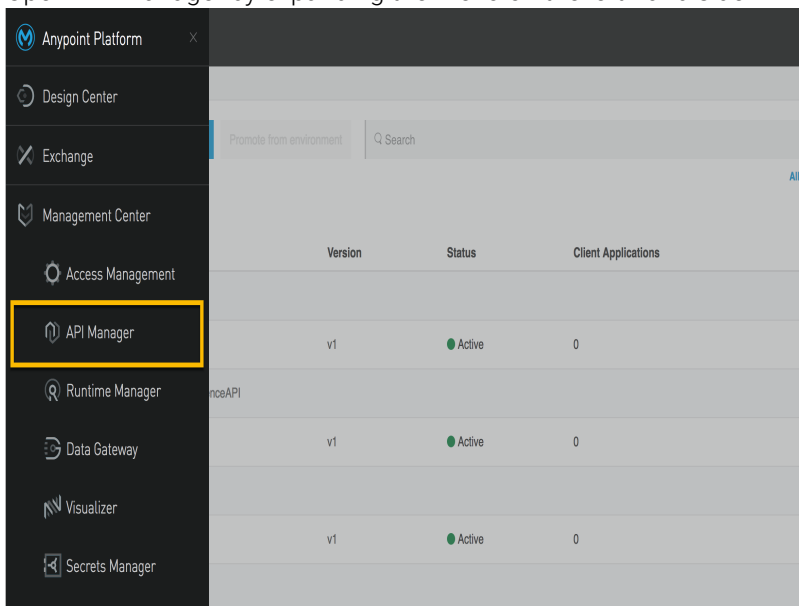
- [PingIntelligence for Mulesoft 3.9](#)
- [PingIntelligence for Mulesoft 4.0](#)

## PingIntelligence for Mulesoft 3.9

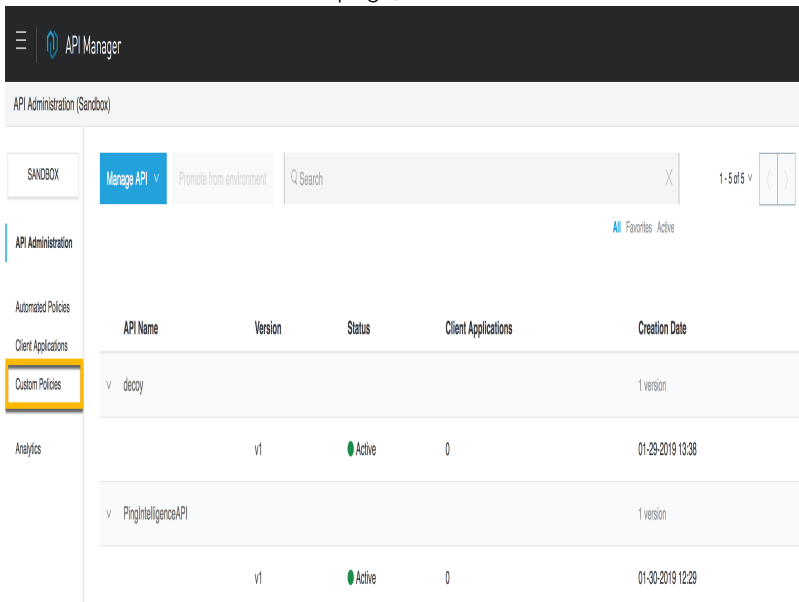
Complete the following steps to deploy the policy. Before applying the PingIntelligence policy, make sure that the API to which you want to apply the policy is defined. The steps mentioned below use an API named `PingIntelligenceAPI` for illustration purpose.

## Deploying PingIntelligence policy to Mulesoft Anypoint

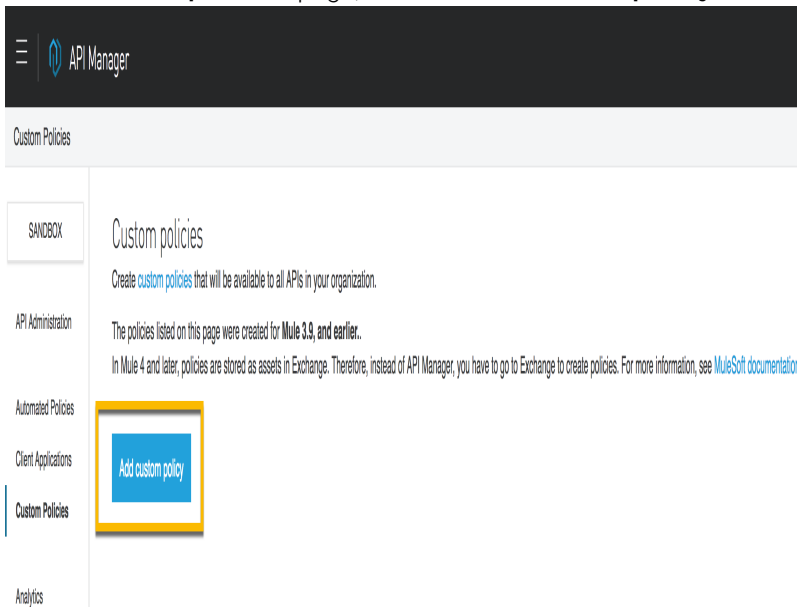
1. Login to your Mulesoft Anypoint account
2. Open API Manager by expanding the menu on the left-hand side:



3. In the **API Administration** page, click on Custom Policies:



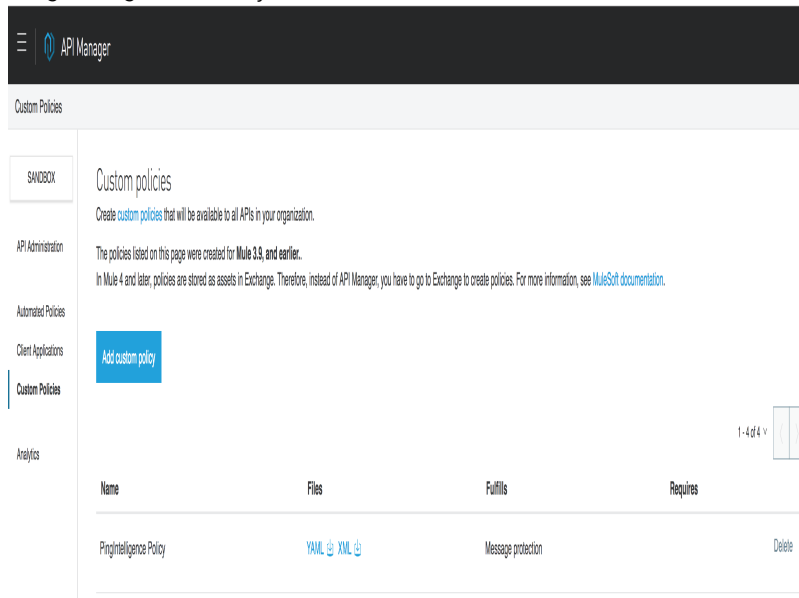
- In the **Custom policies** page, click on **Add custom policy**:



- In the Add custom policy pop-up window, add the policy name, for example, PingIntelligence Policy and upload the `pi_policy.yaml` and `pi_policy.xml` files:

The screenshot shows the 'Add custom policy' dialog box. At the top, it says 'Add custom policy' with a close button (X). Below this, there are two radio buttons under 'Mule Version': 'Policy for runtimes older than Mule 4' (selected) and 'Policy for Mule 4 or later'. The 'Name' field is required and contains the text 'PingIntelligence Policy'. Below the name field, there is a description: 'A YAML file defining the configuration parameters for the policy.' and a file selection area with a 'Choose File' button and the text 'No file chosen'. Similarly, there is a 'Policy configuration' section with a description: 'The Mule configuration XML for the policy implementation. Learn more [here](#).' and another file selection area with a 'Choose File' button and 'No file chosen'. At the bottom right, there are 'Cancel' and 'Add' buttons.

PingIntelligence Policy is added as shown below:



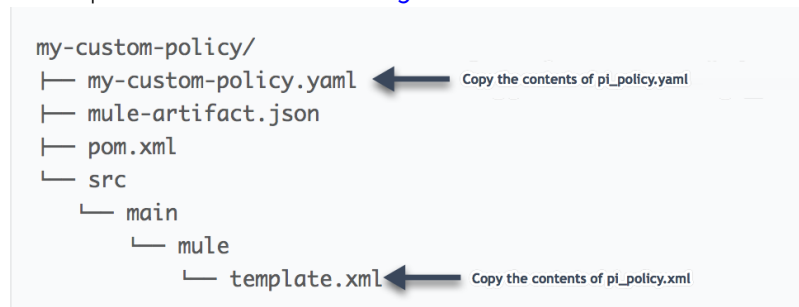
**Parent topic:** [Deploy PingIntelligence policy](#)

## PingIntelligence for Mulesoft 4.0

To deploy the PingIntelligence policy for Mulesoft 4.0, you need to upload the policy to Exchange. Before applying the PingIntelligence policy, make sure that the API to which you want to apply the policy is defined.

Follow the steps mentioned at [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](#):



After the project is set up, complete the steps mentioned in the following two links to upload the PingIntelligence policy:

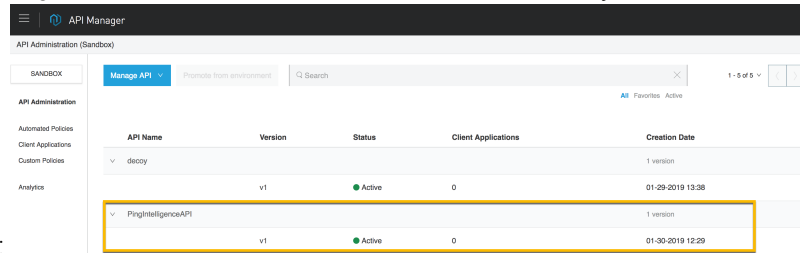
- [Packaging a Custom Policy](#)
- [Uploading a Custom Policy to Exchange](#)

**Parent topic:** [Deploy PingIntelligence policy](#)

## Apply PingIntelligence policy

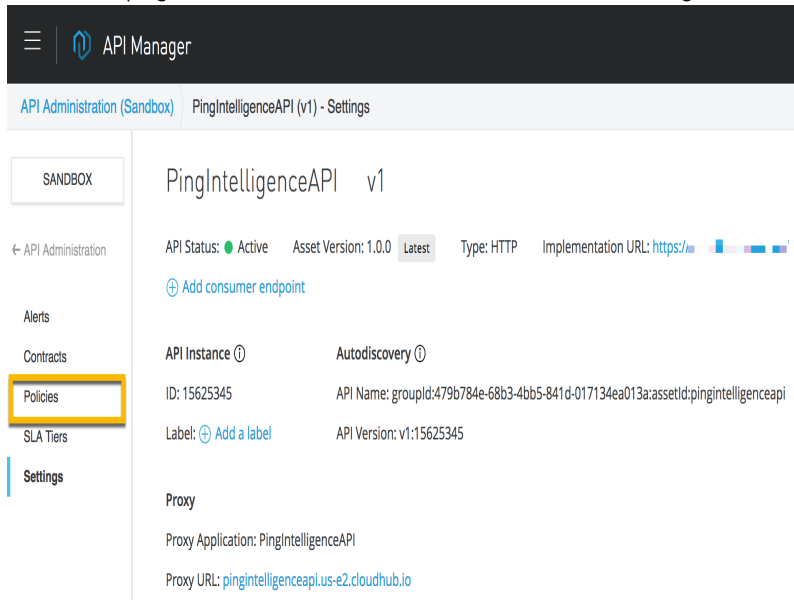
Complete the following steps to attach PingIntelligence policy to your API:

1. Navigate to the API manager and click on the **Version** of the API to which you want to attach the

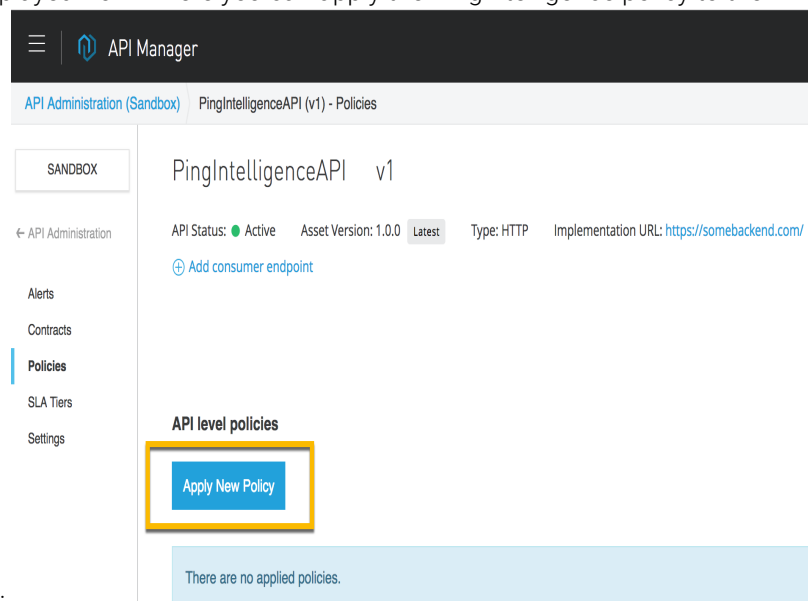


PingIntelligence policy:

2. In the API page, click on **Policies** as shown in the following illustration:



The Policies page is displayed from where you can apply the PingIntelligence policy to the API. Click



on **Apply New Policy**:



3. In the **Select Policy** pop-up window, select the PingIntelligence Policy and click on **Configure**

Policies	Requires
<input type="radio"/> Client ID enforcement ⓘ	—
<input type="radio"/> HTTP basic authentication ⓘ	Security manager
<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 Custom ⓘ	—

**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 if you want Mulesoft to connect to ASE over HTTPS
  - Check the **Allow self-signed certificate** check-box, if you want Mulesoft to accept self-signed certificate from ASE

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

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 **Redeploy**:

---

Deployment Configuration ▾

Runtime version:  ▾

Proxy application name: ⓘ  .cloudhub.io

[Redeploy](#)

## Next steps - Integrate PingIntelligence into your environment

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*.
- *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

You can discover APIs in your environment using PingIntelligence ABS. For more information on enabling 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 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. Oracle Java 8 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-3.2.1.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.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

gateway.management.password	NA
-----------------------------	----

Following is a sample aad.properties file:

```
Automated API Discovery (AAD)
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+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
ASE REST API secret key
ase.secret_key=OBF:AES:Rs7NPeYGPU0Zku7TANJbwEl2rW7+:v7j6VGWaoMjUNcc4IMAtOMtLL8hUPOLWrq7B
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+lRnatylBFug==: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/UN3Efz0DMA/
kmI=:Az82rlUFftMGpMxF7une1JZUucX1911O2QgKvHD36vU=
```

## 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.
```

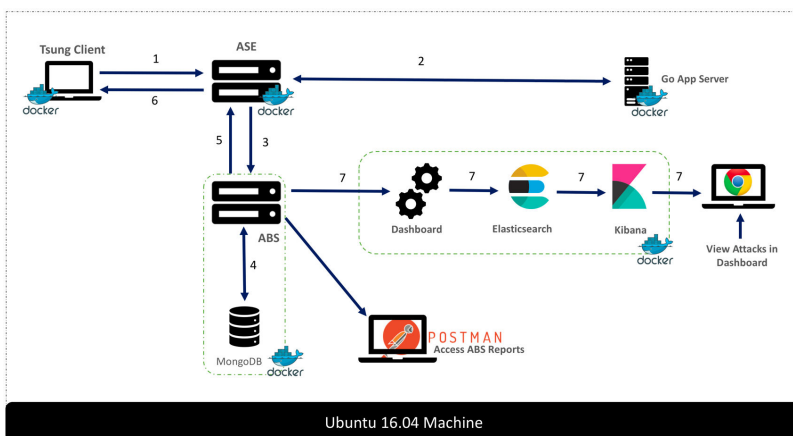
## Docker Inline PoC Deployment

---

### 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](#)
- [Download and untar Docker package](#)
- [Install and load Docker images](#)
- [Setup the PoC environment](#)
- [Start the training](#)
- [Generate sample attacks](#)
- [API deception](#)
- [API discovery](#)
- [View dashboard reports](#)
- [ABS detailed reporting](#)
- [Shutdown the PoC environment](#)
- [Appendix: Verify the Setup](#)

## 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, 16 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.

**Parent topic:**[Docker Inline PoC setup](#)

## 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:

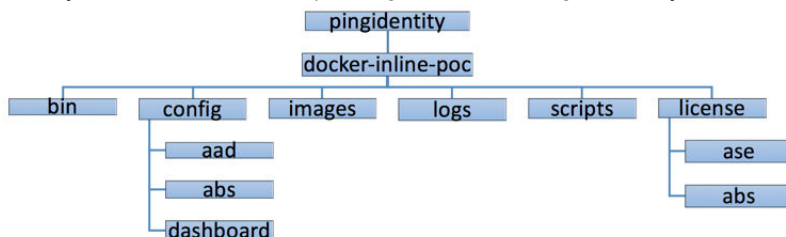
```
$sudo tar -xf /opt/docker-inline-poc.tar.gz
```

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=3.2
IssueDate=2018-11-30
EnforcementType=0
ExpirationDate=2018-12-30
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' 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=3.2
IssueDate=2018-11-30
EnforcementType=0
ExpirationDate=2018-12-30
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/abs` directory, run the following command:

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

**Parent topic:** [Docker Inline PoC setup](#)

## 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 Jan 14 05:30:57 UTC 2019 : loading ASE image
Loaded image: pingidentity/ase:3.2.1
Mon Jan 14 05:31:00 UTC 2019 : loading ABS image
Loaded image: pingidentity/abs:3.2.1
Mon Jan 14 05:31:07 UTC 2019 : loading AAD image
Loaded image: pingidentity/aad:3.2.1
Mon Jan 14 05:31:11 UTC 2019 : loading Dashboard image
Loaded image: pingidentity/dashboard:3.2.1
Mon Jan 14 05:31:25 UTC 2019 : loading client image
Loaded image: pingidentity/client:3.2.1
Mon Jan 14 05:31:29 UTC 2019 : loading server image
Loaded image: pingidentity/server:3.2.1
Mon Jan 14 05:31:32 UTC 2019 : loading mongo image
Loaded image: pingidentity/mongo:3.4.6
Mon Jan 14 05:31:37 UTC 2019 : Installation completed successfully
```

**Parent topic:** [Docker Inline PoC setup](#)

## 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 Jan 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 Jan 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
Sat Jan 5 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 directorySat Jan 5 16:55:13 UTC 2019 : Error : Error during
setup
```

Parent topic:[Docker Inline PoC 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
Mon Dec 10 13:44:25 IST 2018 : Starting model training scripts
Mon Dec 10 13:44:25 IST 2018 : Model training started. Wait 85 minutes for
the model training to complete.
```

Parent topic:[Docker Inline PoC setup](#)

### 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
Mon Dec 10 15:40:51 IST 2018 : Starting attack scripts
Mon Dec 10 16:40:51 IST 2018 : Attack started.
```

Parent topic:[Docker Inline PoC setup](#)

### 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

```

Parent topic:[Docker Inline PoC setup](#)

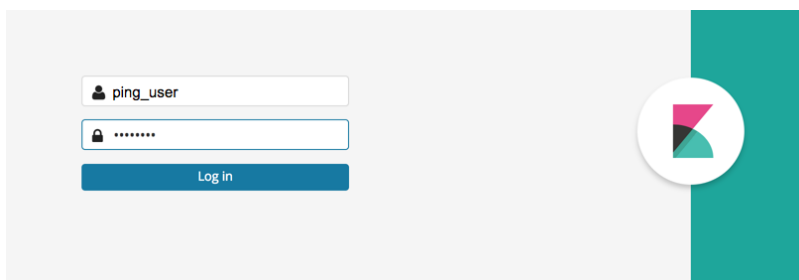
## API discovery

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](#).

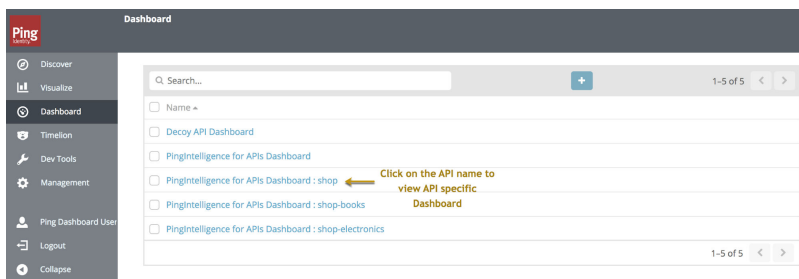
Parent topic:[Docker Inline PoC setup](#)

## 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/changetme

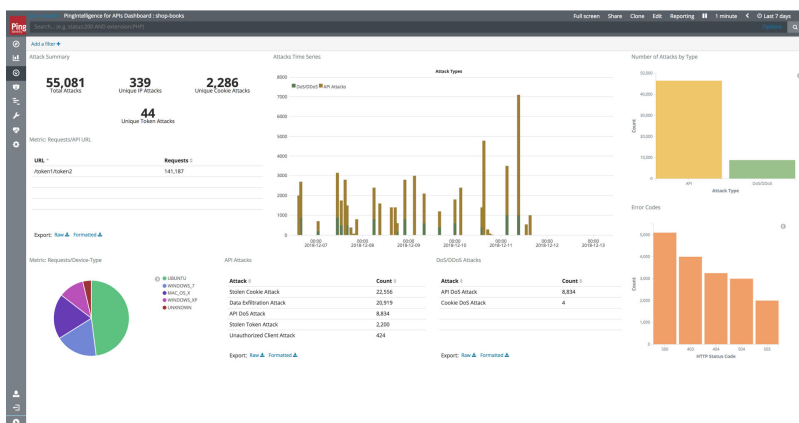


To navigate to the main dashboard, click **PingIntelligence for APIs Dashboard**. To navigate to a specific API's dashboard, click **PingIntelligence for APIs Dashboard: <api name>** for example, **PingIntelligence for APIs Dashboard: shop-books**.



The 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:



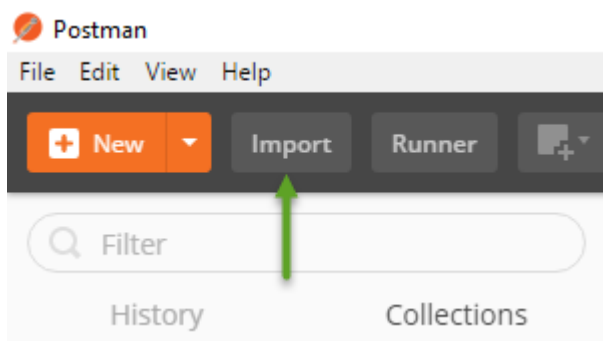
Parent topic: [Docker Inline PoC setup](#)


## 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_3.2.1_Environment` and `ABS_3.2.1_Reports` are files for Postman.
3. Launch the Postman application. Make sure to disable SSL verification in Postman. For more information, see [Using self-signed certificate with Postman](#)
4. Import the downloaded reports files by clicking the **Import** button



5. Click the gear  button in the top right corner
6. In the pop-up window, click **ABS\_3.2.1\_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_a_k` and default Secret\_Key is `abs_s_k`
  - 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**.



## 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`

Parent topic: [Docker Inline PoC setup](#)

# Cloud PoC Service Deployment

---

## 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:

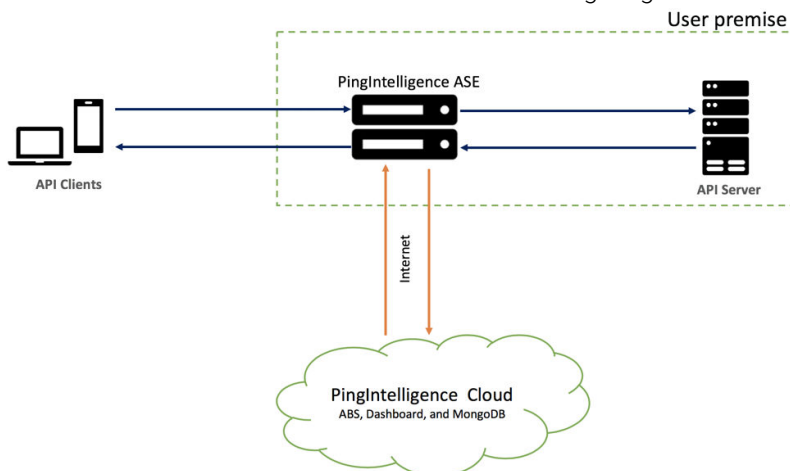
- Inline mode
- Sideband mode

### 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



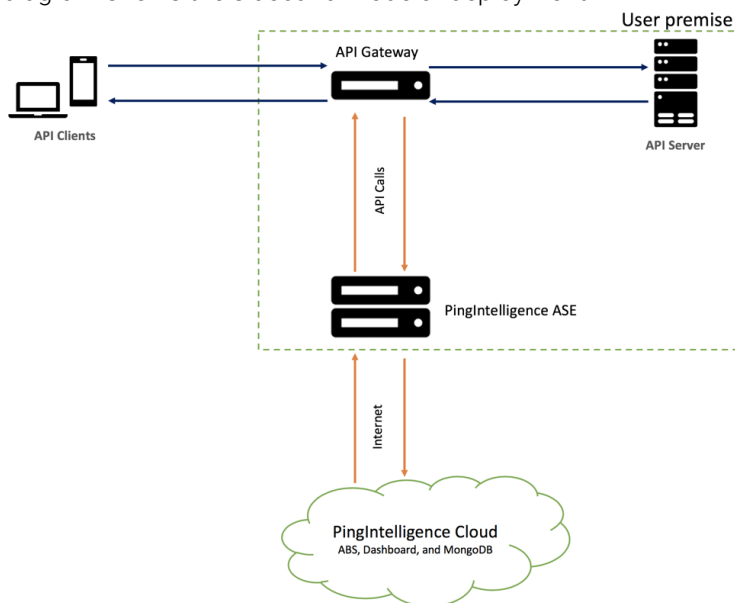
`mode=inline` in the `ase.conf` file. The following diagram shows the inline deployment:



### 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](#)
- [ASE License](#)
- [Obfuscate access and secret key](#)
- [Configure ASE and Dashboard](#)
- [Configure PingIntelligence Cloud Connection](#)
- [Start and stop ASE](#)

- [Enable ASE to ABS engine communication](#)
- [Integrate PingIntelligence into your API environment](#)
- [Add APIs to ASE](#)
- [Download and install AAD](#)
- [AI engine training](#)
- [Connect to the PingIntelligence dashboard](#)
- [Access ABS reporting](#)

## Download and install ASE software

ASE supports RHEL7 or Ubuntu 16 LTS on an EC2 instance, bare metal x86 server, and VMware ESXi. You can install ASE as a root or a non-root user.

Execute the following steps to install ASE:

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-aws-rhel-3.2.1.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-aws-rhel-3.2.1.tar.gz
```

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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.

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## Obfuscate access and secret key

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

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



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

### Obfuscate access and secret keys

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

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

```
Please take a backup of config/ase_master.key, config/ase.conf, config/abs.conf, and config/cluster.conf before proceeding
```

```
If config keys and passwords are already obfuscated using the current master key, they are not obfuscated again
```

```
Following keys will be obfuscated:
```

```
config/ase.conf: sender_password, keystore_password
```

```
config/abs.conf: access_key, secret_key
```

```
config/cluster.conf: cluster_secret_key
```

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

```
obfuscating config/ase.conf, success
```

```
obfuscating config/abs.conf, success
```

```
obfuscating config/cluster.conf, success
```

Start ASE after keys are obfuscated.



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

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## Configure Cloud Connection

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

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

Here is a sample `abs.conf` file:

```
; API Security Enforcer ABS configuration.
; This file is in the standard .ini format. The comments start with a
; semicolon (;).
; Following configurations are applicable only if ABS is enabled with true.

; a comma-separated list of abs nodes having hostname:port or ipv4:port as
; an address.
abs_endpoint=127.0.0.1:8080

; access key for abs node
access_key=OBF:AES://
ENOzsqOEhDBWLDY+pIoQ: jN6wfLiHTTd3oVNzvtXuAaOG34c4JBD4XZHgFCaHry0

; secret key for abs node
secret_key=OBF:AES:Y2DadCU4JFZp3bx8EhnOiw: zzi77GIFF5xkQJccjIrIVWU+RY5CxUhp3NLcNBel+3Q

; Setting this value to true will enable encrypted communication with ABS.
enable_ssl=true

; Configure the location of ABS's trusted CA certificates. If empty, ABS's
; certificate
; will not be verified
abs_ca_cert_path=
```

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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 3.2.1...
please see /opt/pingidentity/ase/logs/controller.log for more details
```

## Stop ASE

Change working directory to bin and run the stop.sh script.

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

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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
```

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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.

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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
<code>protocol</code>	API request type with supported values of: ws - WebSocket ; http - HTTP
<code>url</code>	The value of the URL for the managed API. You can configure up to the "/shopping"- name of a 1 level API "/shopping/electronics/phones" – 3 level API "/" – entire server (used for ABS API Discovery or load balancing)
<code>hostname</code>	Hostname for the API. The value cannot be empty. "*" matches any hostname.
<code>cookie</code>	Name of cookie used by the backend servers.
<code>oauth2_access_token</code>	When true, ASE captures OAuth2 Access Tokens. When false, ASE does not look for OAuth2 Tokens. Default value is false. For more information, see <a href="#">Configuring OAuth2 Token</a> .
<code>apikey_qs</code>	When API Key is sent in the query string, ASE uses the specified parameter. For more information, see <a href="#">Configuring API Keys</a> .
<code>apikey_header</code>	When API Key is part of the header field, ASE uses the specified parameter. For more information, see <a href="#">Configuring API Keys</a> .
<code>login_url</code>	Public URL used by a client to connect to the application.
<code>health_check</code>	When true, enable health checking of backend servers. When false, no health checks are performed. Ping Identity recommends setting this parameter as true.
<code>health_check_interval</code>	The interval in seconds at which ASE sends a health check to determine
<code>health_retry_count</code>	The number of times ASE queries the backend server status after not
<code>health_url</code>	The URL used by ASE to check backend server status.

<b>server_ssl</b>	When set to <b>true</b> , ASE connects to the backend API server over SSL.
<b>Servers:</b> <b>host</b> <b>port</b> <b>server_spike_threshold</b> <b>server_connection_quota</b>	The IP address or hostname and port number of each backend server. See <a href="#">REST API Protection from DoS and DDoS</a> for information on options.
The following API Pattern Enforcement parameters only apply when API Firewall is activated	
<b>Flow Control</b> <b>client_spike_threshold</b> <b>server_connection_queueing</b> <b>bytes_in_threshold</b> <b>bytes_out_threshold</b>	ASE flow control ensures that backend API servers are protected from traffic spikes. See <a href="#">WebSocket API Protection from DoS and DDoS</a> for information on options.
<b>protocol_allowed</b>	List of accepted protocols Values can be HTTP, HTTPS, WS, WSS.   <b>Note:</b> When Firewall is enabled, <b>protocol_allowed</b> takes precedence over <b>protocol_blacklist</b> .
<b>methods_allowed</b>	List of accepted REST API methods. Possible values are: GET, POST, PUT, DELETE, HEAD
<b>content_type_allowed</b>	List of content types allowed. Multiple values cannot be listed. For example, <code>text/html, application/javascript</code>
<b>Decoy Config</b> <b>decoy_enabled</b> <b>response_code</b> <b>response_def</b> <b>response_message</b> <b>decoy_subpaths</b>	When <b>decoy_enabled</b> is set to true, decoy sub-paths function as decoy endpoints. <b>response_code</b> is the status code (for example, 200) that ASE returns. <b>response_def</b> is the response definition (for example OK) that ASE returns. <b>response_message</b> is the response message (for example OK) that ASE returns. <b>decoy_subpaths</b> is the list of decoy API sub-paths (for example <code>/api/v1/decoy</code> ). See <a href="#">API deception</a> 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
```

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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](#)

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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](#).

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## Connect to the PingIntelligence dashboard

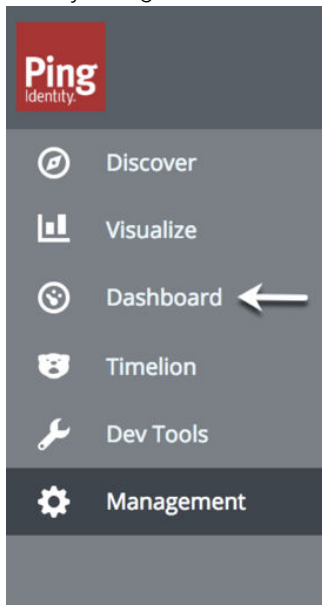
The PingIntelligence Dashboard provides information on the APIs monitored by PingIntelligence for APIs. Until the training period is complete (based on volume of traffic) for an API, only a minimal amount of Dashboard data will be available. If traffic volume is low, it may take several days before many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

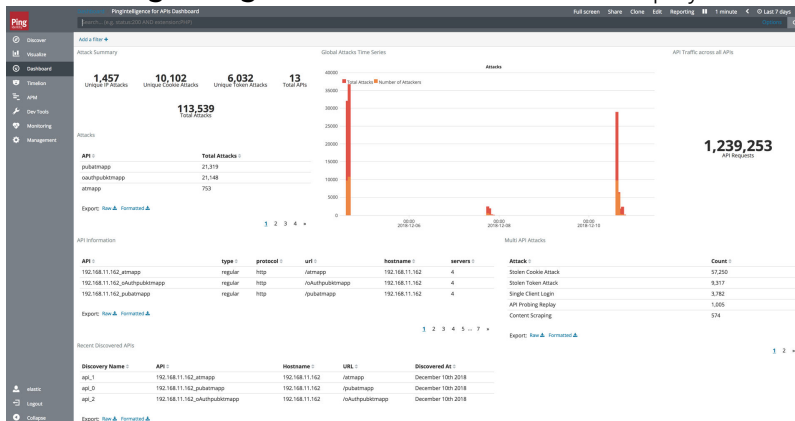
- **Dashboard user and password:** Use this username and password to see the PingIntelligence dashboard.



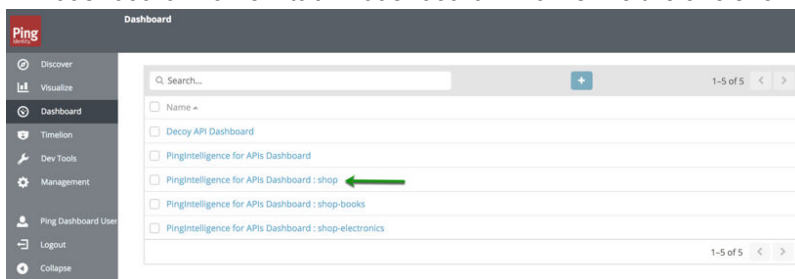
After you log into the Dashboard, click on **Dashboard** to display the list of available Dashboards.



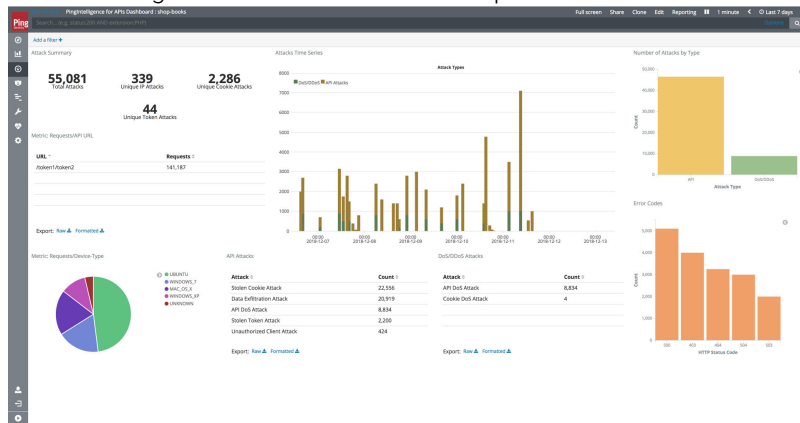
Click on **PingIntelligence for APIs Dashboard** to display the main dashboard.



Click on a listed API name to display the detailed graphs. You can open more than one API by opening each API dashboard in a new tab. A dashboard which is like the one shown below is displayed.



The following screen shot shows an API specific dashboard:



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 unique IP addresses and unique cookies generating attacks. Note: a single IP or cookie could generate more than one attack, so the sum of the unique IPs and cookies 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

### Error and traffic control reporting

- **Server error codes** – Count of each error code returned from API servers.
- **DoS/DDoS threshold exceeded per API** – Count of traffic thresholds exceeded including Server Spike, Client Spike, and Connection Quota Exceeded (See API Security Enforcer Admin Guide for parameters).
- **Blocked connections** – Count of each blocked connection type.

Parent topic: [PingIntelligence Cloud PoC Service](#)

## 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

**Parent topic:** [PingIntelligence Cloud PoC Service](#)

## 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.

**Prerequisites:** [Download](#) the following PingIntelligence products and tools. You also need to download few third-party products.

- **Download products:**
  - PingIntelligence ASE 3.2.1
  - PingIntelligence ABS 3.2.1
  - PingIntelligence AAD 3.2.1
  - PingIntelligence Dashboard 3.2.1

- MongoDB
- JDK 8 update 161 or higher
- Kibana 6.4.3
- Elasticsearch 6.4.3
- **License** Obtain valid ASE and ABS license files from the PingIntelligence sales team.



**Note:**

- Make sure to download the correct ASE binary (RHEL7 or Ubuntu 16 LTS) based on the base image you want to create.
- Download the correct MongoDB binary (RHEL7 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-3.2.1.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</li> <li>• Elasticsearch 6.4.3</li> <li>• Kibana 6.4.3</li> <li>• JDK 8 update 161 or higher</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 ase and abs directory. If you build the Docker images without the license file in ase and abs directory, then you need to map or mount the license file in the following exact location:</p> <ul style="list-style-type: none"> <li>• ASE: /opt/pingidentity/ase/config/ PingIntelligence.lic</li> <li>• ABS: /opt/pingidentity/abs/config/ PingIntelligence.lic</li> </ul>
logs	Contains the log files
software	Contains PingIntelligence ASE, ABS, and AAD

### 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=ubuntu:16.04
```

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

```
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
- MongoDB

It is a good practice to obfuscate the various keys and password in ASE, ABS, AAD 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*
- 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>

3. Download JDK 8 update 161 or later, Kibana 6.4.3, Elasticsearch 6.4.3 and MongoDB 3.4.6 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>
JDK 8	<code>jdk8.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:3.2.1
Image saved to /home/ubuntu/docker-setup/images/pingidentity_ase.tar
Creating build context for abs
Creating Image
Image created with tag pingidentity/abs:3.2.1
Image saved to /home/ubuntu/docker-setup/images/pingidentity_abs.tar
Creating build context for aad
Creating Image
Image created with tag pingidentity/aad:3.2.1
Image saved to /home/ubuntu/docker-setup/images/pingidentity_aad.tar
Creating build context for dashboard
Creating Image
Image created with tag pingidentity/dashboard:3.2.1
```

```
Image saved to /home/ubuntu/docker-setup/images/
pingidentity_dashboard.tar
Creating build context for mongo
Creating Image
Image created with tag pingidentity/mongo:3.4.6
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, 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
CREATED SIZE
pingidentity/dashboard 3.2.1 b172c856756a
32 minutes ago 1.24GB
pingidentity/ase 3.2.1 a29b708d9467
2 hours ago 329MB
pingidentity/mongo 3.4.6 0460e8717341 47
hours ago 485MB
pingidentity/aad 3.2.1 04ed26b185cd
47 hours ago 589MB
pingidentity/abs 3.2.1 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 Jan 18 18:39 ..
-rw----- 1 root root 782182400 Jan 21 10:18 pingidentity_abs.tar
-rw----- 1 root root 600428544 Jan 21 10:18 pingidentity_aad.tar
-rw----- 1 root root 495038976 Jan 21 10:20 pingidentity_mongo.tar
-rw----- 1 root root 339437568 Jan 23 06:57 pingidentity_ase.tar
-rw----- 1 root root 1302484480 Jan 23 08:08
pingidentity_dashboard.tar
drwxr-xr-x 2 root root 4096 Jan 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 true to enable ASE cluster.
ASE_ENABLE_ABS	true/false	Set the value to true 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
ASE_ABS_ACCESS_KEY	string	Access key to connect to ABS
ASE_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	integer	The attack training period
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	integer	The initial period set in hours in which ABS has to be discover APIs. It is a good practice to keep the API discovery interval period less than the initial attack training interval.
API_DISCOVERY_UPDATE_INTERVAL	integer	The time period in hours in which ABS reports the newly discovered APIs
API_DISCOVERY_SUBPATH	integer	The number of subpaths that are discovered in an API. The maximum value is 3.
WIRED_TIGER_CACHE_SIZE_GB	float	Memory in GB to be used by MongoDB cache.

**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
ABS_RESTRICTED_USE_ACCESS	true/false	Set to true 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	
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	string	Gateway management URL. Configure if you have Axway API gateway in your deployment.
GATEWAY_MANAGEMENT_USERNAME	string	Gateway management username. Configure if you have Axway API gateway in your deployment.
GATEWAY_MANAGEMENT_PASSWORD	string	Gateway management password. Configure if you have Axway API gateway in your deployment.
PINGACCESS_MANAGEMENT_URL	string	PingAccess management URL. Configure when mode is set to pa .
PINGACCESS_MANAGEMENT_USERNAME	string	PingAccess management username. Configure when mode is set to pa .
PINGACCESS_MANAGEMENT_PASSWORD	string	PingAccess management password. Configure when mode is set to pa .

## 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

## 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" pingidentity/
mongo:3.4.6
```

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"
pingidentity/abs:3.2.1
```

**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_ENABLE_ABS="true" -e
ASE_ABS_ENDPOINT="abs:8080" -e\
ASE_ABS_ACCESS_KEY="new_abs_ak" -e ASE_ABS_SECRET_KEY="new_abs_sk" --shm-
size=1g \
pingidentity/ase:3.2.1
```

**Launch the second ASE node in ASE cluster:** Run the following command with some sample environment variables to launch the ASE 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
ASE_ABS_ACCESS_KEY="new_abs_ak" -e\
ASE_ABS_SECRET_KEY="new_abs_sk" --shm-size=1g pingidentity/ase:3.2.1
```

**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" ABS_HOST="abs" -e ABS_ACCESS_KEY="new_abs_ak" -e
-e\
ABS_SECRET_KEY="new_abs_sk" -e ASE_HOST="ase" -e\
ASE_ACCESS_KEY="admin" -e ASE_SECRET_KEY="admin" pingidentity/aad:3.2.1
```

**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:3.2.1
```

### 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