



Illumio Core[®]

Version 21.1

REST API Developer Guide

November 2022

10000-100-21.1

Legal Notices

Copyright © 2021 Illumio 920 De Guigne Drive, Sunnyvale, CA 94085. All rights reserved.

The content in this documentation is provided for informational purposes only and is provided "as is," without warranty of any kind, expressed or implied of Illumio. The content in this documentation is subject to change without notice.

Product Version

PCE Version: 21.1 (Standard Release)

For the complete list of Illumio Core components compatible with Core PCE, see the Illumio Support portal (login required).

For information on Illumio software support for Standard and LTS releases, see [Versions and Releases](#) on the Illumio Support portal.

Resources

Legal information, see <https://www.illumio.com/legal-information>

Trademarks statements, see <https://www.illumio.com/trademarks>

Patent statements, see <https://www.illumio.com/patents>

License statements, see <https://www.illumio.com/eula>

Open source software utilized by the Illumio Core and their licenses, see [Open Source Licensing Disclosures](#)

Contact Information

To contact Illumio, go to <https://www.illumio.com/contact-us>

To contact the Illumio legal team, email us at legal@illumio.com

To contact the Illumio documentation team, email us at doc-feedback@illumio.com

Contents

Chapter 1 Overview of the Illumio REST API	12
API Classification and Version	12
Public Stable APIs	12
Public Experimental APIs	13
Private APIs	13
Illumio REST API Versions	13
Illumio REST API Schema Files	13
REST API URIs	13
API Version and Organization HREF	13
Port Number	14
GET Collections URI Syntax	15
Non-GET Collections URI Syntax	15
Security Policy Items and “:pversion”	16
REST API Limits	16
API Rate Limits and DOS Protection	16
Request Rate and Item Limits for Bulk Operations	16
Ruleset Rules Display Limit	17
GET Collection Request Limits	17
Checking Total Item Count	17
Character Limits on Resource Names	18
HTTP Requests and Responses	19
HTTP Request Headers	19
HTTP Request Body	19
PUT Operations	19
Response Header Request-ID	19
Response Types	20
Request Calls Using Curl	22
Curl Overview	22
Curl-specific Options	23
Using Curl with json-query	24
Chapter 2 Authentication and API User Permissions	25
Permissions Required for API Users	25
User Permissions and the API	26
Unscoped Roles	26

Scoped Roles	26
Session Credentials	27
About Session Credentials and Tokens	27
Authenticate to Login Service	28
Create Session Credentials Using Login API	30
API Keys	32
About API Keys	32
API Key Methods	33
Get API Keys	33
Create an API Key	36
Create a Key Using the PCE Web Console	38
Update an API Key	39
Delete an API Key	39
REST API Users	40
Users API Methods	40
Log Into the PCE	40
Log Out and Destroy Session Credentials	41
Get User Information	41
Create a New User	43
Update User Information	44
Change the User Password	45
LDAP Authentication	46
Prerequisites and Limitations	46
Configuring LDAP Authentication for the PCE	47
How to set up the PCE for LDAP Authentication	49
Use Cases	56
REST API Schema Files	59
Sample Responses	60
Chapter 3 Asynchronous GET Collections	65
Overview of Async GET Requests	65
REST Patterns for Collection vs. Instance	65
Async GET Supported APIs	66
Async Job Operations	68
Workflow for Async Job Operations	68
Create an Async Job Request	69
Poll the Job	69

Get Async Job Results	71
Poll the Query Job Status	72
Delete a Job	74
Get the Job Results	74
Chapter 4 PCE Management	76
<hr/>	
Product Version	76
Authentication Settings	77
Authentication Settings API Methods	77
Get Authentication Settings	77
Update Authentication Settings	78
Password Policy	78
Password Policy API Methods	78
Update the Password Policy	80
Supercluster Leader	82
About the Supercluster Leader API	82
Get Supercluster Leader	82
PCE Health	82
About the PCE Health API	83
PCE Health API Method	83
Check PCE Health	83
Node Availability	96
No Op	97
Events	97
Event Types	98
Event API Methods	98
Get Events	98
Get Events Collection	98
Organization Settings	102
Get Events Settings	102
Update Events Settings	102
Syslog Destinations	102
Create a Syslog Destination	105
Update a Syslog Destination	106
Delete a Syslog Destination	107
Container Clusters	107
Container Cluster API	107

Container Cluster Workload Profiles API	113
Label Restrictions	120
Container Cluster Service Backend API	123
Access Restrictions and Trusted Proxy IPs	125
Access Restrictions	125
Trusted Proxy IPs	128
Chapter 5 Provisioning	131
<hr/>	
Policy Update Mode	131
Overview of Policy Update Mode	131
Policy Update Mode Methods	132
Get Policy Update Mode	132
Change Policy Update Mode	134
Remove all Static Policy Scopes	135
Provisioning (public stable)	136
Provisioning API Methods	136
Provision All Items	136
Provision Individual Items	137
Restore the Previous Security Policy	139
Get All Provision Versions	139
Get an Individual Provision Version	140
Provisioning (public experimental)	141
Provisioning API Methods	142
Provisionable Policy Items	142
Policy Provisioning States	143
Get All Items Pending Provisioning	143
Revert All Items Pending Provisioning	144
Revert a List of Items Pending Provisioning	144
Get Provisioning Dependencies	145
Get Modified Items in a Provisioned Version	149
Get Rules Allowing Communication	150
Selective Enforcement	152
Selective Enforcement Methods	153
Virtual Server Filtering	158
Virtual Server Endpoints	159
New Filters for Virtual Servers	159
Schema Changes for Virtual Server Endpoints	160

Request and Response Examples	161
Chapter 6 RBAC for PCE Users	166
<hr/>	
RBAC Overview	166
RBAC Terms and Concepts	167
Grant Permissions Workflow	168
RBAC User Operations	168
RBAC Users API Methods	169
Get RBAC Users	169
Create a Local User	171
Convert Local User to External User	172
Convert External User to Local User	173
Re-invite a Local User	173
Authorization Security Principals	174
Authorization Security Principals API Methods	174
Get Authorization Security Principals	174
Create an Authorization Security Principal	176
Update an Authorization Security Principal	177
Delete an Authorization Security Principal	178
RBAC Permissions	178
RBAC Permissions API Methods	179
Get RBAC Permissions	180
Grant RBAC Permissions	182
Update an RBAC Permission	184
Delete an RBAC Permission	185
Organization-wide Default User Permissions	185
About Default User Permissions	185
Get a Collection of Authorization Security Principals	185
Get Permission for the Null Auth Security Principal	186
Delete Null Authorization Security Principal Permission	187
Re-Enable the Organization Read-Only Permission	187
App Owner RBAC Role	188
App Owner Roles	189
Chapter 7 Rulesets and Rules	190
<hr/>	
Rulesets	191
Ruleset API Methods	191
Active vs Draft	191

Ruleset Components	191
Example Ruleset Scope	192
Ruleset Rules	193
Get Rulesets	193
Create a Ruleset	197
Update a Ruleset	199
Delete a Ruleset	200
Rules	200
Rules API Methods	200
Active vs Draft	200
Rule Types	201
Rule Type JSON Specification	201
Providers and Consumers	202
Stateless Rules	203
Get Rules	203
Create Rules	206
Update Rules	214
Delete a Rule	214
Rule Search	214
Custom iptables Rules	217
Terminology: Custom iptables Rules	218
How Custom iptables Rules Work	218
Create a Custom iptables Rule	219
Machine Authentication	225
Configure Machine Authentication	225
Configure Machine Authentication on Rule	227
Chapter 8 Security Policy Objects	229
Overview of Security Policy Objects	230
Active vs. Draft	230
Labels	230
Labels API Methods	230
Get Labels	230
Create a Label	234
Update a Label	236
Delete a Label	237
Label Groups	237

Label Groups API Methods	238
Active vs. Draft	238
Get Collection of Label Groups	238
Update a Label Group	242
Delete a Label Group	243
Services	243
Services API Methods	243
Active vs. Draft	243
Get Services	244
Create a Service	248
Update a Service	249
Delete a Service	250
Virtual Services and Service Bindings	251
Virtual Services	251
Create an Individual Virtual Service	258
Service Bindings	262
IP Lists	267
IP Lists API	268
Active vs Draft	268
Get IP Lists	268
Create an IP List	271
Update an IP List	273
Delete an IP List	273
Virtual Servers and Load Balancers	274
Load Balancers	274
Virtual Servers	275
Security Principals	280
Security Principals API Methods	280
Query Parameters for GET	280
Query Parameters for POST	281
Query Parameters for PUT and DELETE	282
Chapter 9 Visualization	285
Vulnerabilities	285
Vulnerabilities API Methods	286
Get Collection of all Vulnerabilities	286
Get a Vulnerability	287

Create or Update a Vulnerability	288
Delete a Vulnerability	289
Vulnerability Reports	289
Vulnerability Reports API Methods	290
Get a Collection of Vulnerability Reports	290
Get a Vulnerability Report	291
Create or Update a Vulnerability Report	292
Delete a Vulnerability Report	294
Explorer	294
Explorer API Method	294
Example Explorer Search	299
Bulk Traffic Loader	307
Bulk Traffic Loader API Methods	307
Workflow to Upload Bulk Traffic	307
Create Collection of Unmanaged Workloads	308
Upload Bulk Traffic Flows	314
Chapter 10 Workloads	317
<hr/>	
Pairing Profiles and Pairing Keys	318
About Pairing Profiles and Keys	318
Pairing Profile Methods	318
Get Pairing Profiles	318
Create a Pairing Profile	324
Update a Pairing Profile	326
Delete a Pairing Profile	326
Pairing Key API Method	326
Create a Pairing Key	327
Workload Operations	327
Update Workload Information	327
Unpair Workloads	332
Mark Workload as Suspended	334
Create an Unmanaged Workload	335
Delete a Workload Record	339
Workload Interfaces	340
Workload Network Interfaces API Methods	340
Get Workload HREF and Interface Names	340
Get Workload Network Interface	341

Create Workload Network Interface	342
Delete Workload Network Interface	343
Workload Settings	344
Get Workload Settings	344
Update Workload Settings	345
Workload Bulk Operations	347
About Bulk Operations	347
Workload Bulk Operations Methods	347
Caveats for Workload Bulk Operations	347
External Data Reference IDs for Workloads	348
Create a Collection of Workloads	348
Update Collection of Workloads	350
Delete a Collection of Workloads	351
VEN Operations	353
Overview of VEN Suspension	353
VEN API Methods	354
Agents on Workloads	358
Agents API Methods	358
Get an Individual Agent	358
Update an Agent	361
Blocked Traffic to and from Workloads	362
Filtering and Aggregating Traffic	362
Traffic Collector API Methods	363

Overview of the Illumio REST API

This chapter contains the following topics:

API Classification and Version	12
REST API URIs	13
REST API Limits	16
HTTP Requests and Responses	19
Request Calls Using Curl	22

The Illumio API is a RESTful API and uses JSON over HTTPS. JSON is used to encode all data transfer in both directions, so that everything sent to and everything received from the API gets encoded in JSON.

To work with Illumio API, you need to be authorized by an Illumio administrator and to have the appropriate credentials for authentication.

API Classification and Version

This chapter explains the distinction among the Illumio Public Stable, Public Experimental, and private APIs.

Public Stable APIs

The Public Stable APIs are generally available to all Illumio customers, are documented, and are stable. “Stable” means that Illumio will not introduce any further breaking changes to the API. If a breaking change is required, another version of the API will be introduced, and the previous version will continue to be supported for a minimum of six (6) months.

Public Experimental APIs

The Public Experimental APIs are generally available to all Illumio customers, are documented, but are subject to change from release to release. If you use experimental APIs, such as in scripts, be aware that some of them might change. Some of these APIs might be promoted to Public Stable at a future date, or could be made no longer available.

To help distinguish which APIs are "Public Experimental," this API guide uses orange color for headings inside these files.

Private APIs

In addition to the Public Stable or Public Experimental APIs, the Illumio Core includes additional Private APIs used by the PCE web console. The private Illumio APIs are not exposed to end-users, are not documented, or supported for use.

Illumio REST API Versions

Illumio REST APIs follow the release versions of other Illumio components, such as the PCE and VEN.

Illumio REST API Schema Files

Illumio REST API schema files follow the standard JSON schema form described at <http://json-schema.org/>. The file name convention is the Illumio REST API URL name with underscore rather than slashes + `_` + operation + `.schema.json`. For example, for the login API, the payload schema file is named: `user_login_get.schema.json`.

REST API URIs

This section describes the URI syntax used with this API, which can be different depending on the REST call you are making and the types of Illumio resources on which you are operating.

API Version and Organization HREF

The API version and organization HREF are two variables used in every call made to this API.

The current version of the Illumio Core REST API is version 2 (v2), which is represented in method URIs by the `[api_version]` variable. Version 1 (v1) is still supported.

**NOTE:**

The parameter tables and code examples in this document typically describe the v1 APIs, which in many cases are the same or very similar to the v2 APIs. For v2 API parameter tables, code examples, and authorization permissions, see the [Illumio Core REST API Reference](#).

You can determine the organization HREF for the PCE when you use the login API to authenticate with the PCE and obtain a session token. In method URIs, this value is represented by the [org_href] variable.

In response to using the login API, the organization HREF is listed as shown, but depends on the version of the API you are using:

```
"orgs": [  
  {  
    "org_id": 2,  
    "org_href": "/orgs/2",
```

Note that both [api_version] and [org_href] begin with a forward slash:

- [api_version] - /api/v2
- [org_href] - /orgs/2

For example, to get a collection of labels that exist inside an organization, construct the URI as follows, with the API version and the organization HREF shown in blue font:

```
GET [api_version][org_href]/labels
```

To get all of the API keys created by a specific user, construct the URI as follows, with the HREF path to the user shown in a blue font:

```
GET api/v2/orgs/1/api_keys
```

Port Number

The port number used in the code examples is 8443, which is the default. However, since the port number might be different depending on the implementation, ask your Illumio system administrator which port number to use when making calls to the Illumio Core REST API.

GET Collections URI Syntax

The base URI for Illumio REST API endpoint for GET collections:

```
GET http://[pce_hostname]:[port][api_version][org_href]/[api_endpoint]
```



IMPORTANT:

When making API calls, the `[pce_hostname]` or `[pce_hostname]:[port]` should not end with a forward slash (`/`). This is because `[api_version]` begins with a forward slash.

For example, the URI for getting a collection of workloads uses this syntax:

```
GET https://pce.my-company.com:8443/api/v2/orgs/1/workloads
```

In the rulesets API, you also have the ability to get all of the rules ("`sec_rules`") contained in a ruleset. The URI syntax for this operation is as follows:

```
GET http://[pce_hostname]:[port][api_version][object_href][api_endpoint]
```

For example:

```
GET [api_version][ruleset_href]/sec_rules
```

Non-GET Collections URI Syntax

For the non-GET methods of PUT, POST, and DELETE, the object HREF is listed as the endpoint, as shown here:

```
PUT [api_version][object_href]
```

The relative path of the `[api_version]` ("`api/v2/`") indicates that version 2 of the API is in use.

In the URI above, `[org_href]` is not added because it is included in the `[object_href]` string. For example, this is the `[object_href]` for a Workload:

```
/orgs/2/workloads/3e3e17ce-XXXX-42b4-XXXX-1d4d3328b342
```

Another case is performing PUT, POST, or DELETE operations on the rules contained in a ruleset. The URI syntax is the same as a GET operation.

Security Policy Items and “:pversion”

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, firewall settings, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID of the security policy.

The URI for security policy items is as follows:

```
[pce_host][port][api_version][org_href]/sec_policy/:pversion/[api_endpoint]
```

REST API Limits

When making API calls, make sure that you take into account the allowed maximum number of calls per minute, returned objects, or total item count.

API Rate Limits and DOS Protection

The Illumio REST API is rate-limited and allows only a maximum of 500 requests per minute per user session or API key. The rate is set to maintain the PCE performance and service availability, and to prevent malicious attackers attempting to disrupt a service (for example, DoS attacks). If the set rate limit is reached, the call returns an HTTP error 429 Too many requests.

Request Rate and Item Limits for Bulk Operations

In addition to the rate limits described above that are counted for all requests, the unpair workloads and delete traffic flows APIs have a rate limit of 10 calls per minute. There are also two limits on the number of resources that can be operated on per call.

API Call and Endpoint	Request Rate Limit	Item Limit	Exposure
Unpair Workloads PUT [api_version][org_href]/-workloads/unpair	10 per minute	1000 workloads per request	Public Stable



NOTE:
Illumio reserves the right to adjust the rate limit on the Illumio Secure Cloud for given endpoints at any time to ensure all clients receive a high-quality service.

Ruleset Rules Display Limit

The PCE web console supports up to 500 rules per ruleset. If you need to write more than 500 rules for a particular scope, create additional rulesets or use the Illumio Core REST API. Rulesets with more than 500 rules are not be fully displayed in the PCE web console.

GET Collection Request Limits

By default, when you perform a synchronous GET request with this API, the maximum number of objects returned is 500.

Some GET APIs provide query parameters to help restrict the number of results, depending on the API. For example, the Workloads API provides multiple query parameters for GET collections, such as `label`, `ip_address`, `policy_health`, and more.

If you wish to get more than 500 objects from a GET collection, use an [asynchronous GET collection](#), which runs the request as an offline job. Job results can be downloaded after the job finishes.

Checking Total Item Count

To find out how many items exist for a given resource, such as whether there are more than 500 workloads in the PCE, first check the number of items using the `max_results` query parameter on a GET collection and then view the header of the response for the total item count for the resource.

If the total item count is less than 500, you can perform a regular GET collection for the results. If the total item count is more than 500, use an [asynchronous GET collection](#).

For example, make the following GET call on a collection of workloads with the `max_results` query parameter set equal to 1, then check the header to see how many workloads exist in your organization.

**NOTE:**

When using multiple query parameters, enclose the URI, endpoint, and `query_params` in single quotes or double-quotes.

```
GET 'https://pce.mycompany.com:8443/api/v2/orgs/7/workloads?max_results=1&managed=true'
```

You can check the HTTP response header for the `'X-Total-Count'` field, which indicates the total number of workloads. In this example, the total count shows 71 (highlighted in blue font), so a regular GET collection is appropriate. If the value were more than 500, then an asynchronous GET collection would be used.

```
Cache-Control →no-store
Content-Encoding →gzip
Content-Type →application/json
Date →Wed, 07 Sep 2016 14:01:00 GMT
ETag →W/"025cc8bfcXXXXXXXXXX7900081e7c6cb"
Status →200 OK
Transfer-Encoding →chunked
Vary →Accept-Encoding
X-Matched-Count →71
X-Request-Id →d43a8ce9-XXXX-4453-XXXX-dde79XXX0fa8
X-Total-Count →71
```

Character Limits on Resource Names

When naming resources, the PCE has a 255 character limit for each name string. This JSON property is listed as `'name'` in the API.

For example, this 255 character limit applies when naming such things as workloads, labels, IP lists, and services

However, the PCE does not have a character limit for the description field that typically follows the name of a resource.

HTTP Requests and Responses

This section explains how to formulate HTTP requests and read HTTP responses.

HTTP Request Headers

Set an `Accept: application/json` header on all GET operations (optional for DELETE operations):

```
-H 'Accept: application/json'
```

Set a `Content-Type: application/json` header on PUT and POST operations:

```
-H 'Content-Type: application/json'
```

HTTP Request Body

Most of the parameters and data accompanying requests are contained in the body of the HTTP request. The Illumio REST API accepts JSON in the HTTP request body. No other data format is currently supported.

PUT Operations

Illumio REST API PUT operations modify a subset of attribute-value pairs for a specified resource. The attributes that are not specified in the PUT operation are left unmodified.

For example, to update a user's phone number (using the Users API) without modifying the user's address, call PUT with a request that only modifies the phone number, and only the phone number is changed.

Response Header Request-ID

The Illumio REST API provides a useful troubleshooting feature that returns a unique Request-ID in the HTTP response header on calls made with this API.

You can provide the Request-ID when opening Illumio support tickets, which are designed specifically for operations that produce errors. The Request-ID helps Illumio support to troubleshoot specific operations and errors.

If you are using curl to make REST API calls to the PCE, you can specify the curl `-D` flag plus a file name to write the response header to a file.

The following example shows a curl command to get a collection of workloads that uses the -D flag to write the response header to a file named temp_header.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/workloads -H "Accept: application/json" -u $KEY:$TOKEN -D temp_header
```

The file contains the response header of the call (highlighted in **blue bold font**):

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 16:58:00 GMT
Content-Type: application/json
Content-Length: 2193032
Connection: keep-alive
Vary: Accept-Encoding
Vary: Accept-Encoding
Status: 200 OK
X-Total-Count: 1406
X-Matched-Count: 1406
ETag: "523d67cbd57b18d0e97bc8e7555142eb"
Cache-Control: max-age=0, private, must-revalidate
X-Request-Id:9722c8b5-94dc-4a50-853a-8e8f22266528
Cache-Control: no-store
Pragma: no-cache
```

Response Types

The HTTP response includes:

- An HTTP status code
- A response body that contains data in JSON format:
 - Your requested data if successful
 - An error code and message if there is an error

HTTP Status Codes — Success

The following table lists all expected success codes returned when you use the Illumio REST API:

HTTP Code	Description
200 OK	Successful operation where JSON body is returned
201 Created	Successful POST operation where an object was created

HTTP Code	Description
204 No Content	Operation succeeded and nothing was returned

HTTP Status Codes — Failure

All Illumio REST API methods (GET, POST, PUT, and DELETE) might fail with an error in the 400 range. The error code 400 usually means that either the resource is not available (such as trying to update a previously deleted label), or there is a mistake in the URL (such as specifying `/sh1abe1s` instead of `/1abe1s`).

Other errors that might occur:

HTTP Code	Description
400 Bad Request	Something in the curl request was not correct, for example "curl -X -i GET" instead of "curl -i -X GET"
401 Authentication failure or HTTP/1.1 401 Unauthorized	For example, the user attempted to make an API call but forgot to log in, username or password were incorrect or missing, or a missing space before "-u"
403 Authorization failure	For example, the user is not authorized to make the call.
HTTP/1.1 403 Forbidden	For example, using the incorrect HTTP method (like using GET instead of POST), the incorrect <code>org_id</code> parameter was used
404 Invalid URL	
HTTP/1.1 404 Not Found	For example, an incorrect API version number <code>/api/v191/</code> , missing or incorrect <code>org_id</code> , <code>/orgs/{org_id}/</code> , wrong URL, or a misspelled endpoint.
404 Page not found	For example, the wrong <code>org_id</code> in the URI or missing blank space before an option dash, like before <code>-H 'Accept: application/json'</code>
405 Method not allowed	For example, if you are performing a POST on a resource that only allows PUT.
406 Invalid payload	The JSON request payload was constructed improperly.

Other Failure Codes

```
-bash: -H: command not found HTTP/1.1 401 Unauthorized
```

- This can be caused if more than one query parameter is used and the URI (including the query parameters) is not enclosed with single quotes or double quotes.
Example:
`'https://pce.my-company.com:8443/api/v2/orgs/2//workloads?managed=true&max_results=1'`

curl: (3) Illegal port number

- For example, a missing blank space between `-u uname:'pswd'` and the next option, for example `-H 'Accept: application/json'`.

parse error: Invalid numeric literal at line 1, column 9

- Can be caused by an incorrect curl command, for example including a path parameter that isn't allowed, like using `orgs/org_id` for an endpoint that doesn't use it. This is also a known JSON query bug caused by using `-i` in a curl command that uses `json-query`. To see the headers returned from the curl command, remove `json-query` from the curl command and use `-i`, for example `"curl -i -X GET ..."`

curl: (23) Failed writing body

- Can be caused by calling an endpoint that doesn't exist.

The property '#/' of type null did not match the following type: object in xxxxxxx.schema.json

- Can be caused by a missing or incomplete request body.

```
[{"token": "input_validation_error", "message": "Input validation failed. Details: {The property '#/' of type NilClass did not match the following type: object in schema xxxxx.schema.json}"}]
```

- Is the wrong `-H` value being used? For example, is `-H 'Accept: application/json'` being used for a PUT or a POST instead of `-H 'Content-Type: application/json'?`

Request Calls Using Curl

This section explains how to use curl commands to work with Illumio APIs by defining some standard options and constants.

Curl Overview

Curl is a common command-line data transfer tool for making API calls and is especially useful in scripts written for automated tasks.

The syntax for using curl with the API for logging a user into the PCE is as follows:

```
curl -i -X <HTTP method> <uri_of_api> <header> -u $KEY:$TOKEN -Options
```

The syntax for using curl with the API for PUT operations using an API key for authentication is as follows:

```
curl -i -X PUT <URI of API> -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{ "json_property": "property_value", "json_property": "property_value" }'
```

For example:

```
curl -i -X PUT https://scp.illum.io:8443/api/v2/users/11/local_profile/password -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{ "current_password": "NotMyReal_Old*96Password", "new_password": "NotMy*76New!pswd" }'
```

Curl-specific Options

For the curl examples provided in this API documentation, a few standard curl options are defined as follows.

The user and password to use for server authentication:

```
-u/--user <user:password>
```

For brevity, code examples typically use constants for `-u username:'password'` arguments. `$TOKEN` represents an authentication token (a string enclosed by single quotes to prevent it from unintentionally expanding):

```
-u $KEY:$TOKEN
```

(HTTP) Header to use when getting a web page:

```
-H/--header <header>
```

(HTTP) Specify a the HTTP method to use when communicating with the HTTP server:

```
-X/--request <command>
```

Example:

```
-X POST
```

(HTTP) Send the specified data in a POST request to the HTTP server in a way that emulates a user filling in an HTML form, and clicking **Submit**:

```
-d/--data <data>
```

Example API Call Using CURL

To get all of the API keys of a specific user using the user's session credentials:

```
curl -i -X GET https://scp.illum.io:8443/api/v2/users/11/api_keys -H "Accept: application/json" -u $KEY:$TOKEN
```

Using Curl with json-query

When using json-query to format the output of curl commands, be aware that due to a json-query bug, this does not work with the curl -i option, which displays response headers. When you use the curl -i option, such as to see the total number of workloads when using GET workloads, you might get various error messages like curl: (3) Illegal port number. To work around this issue, remove the -i option and retry the curl command.

Chapter 2

Authentication and API User Permissions

This chapter contains the following topics:

Permissions Required for API Users	25
Session Credentials	27
API Keys	32
REST API Users	40
LDAP Authentication	46

To use the REST APIs, you must be an authorized Illumio user and have credentials to log into the PCE.

You get authorized to perform a specific job according to the privileges granted to you based on the role-based access control (RBAC) and implemented by the Illumio administrator.

The PCE has two types of credentials that you can use to authenticate with it and make REST API calls:

- API keys, which provide a persistent means of authenticating
- Session credentials, which provide a temporary means of authenticating

Permissions Required for API Users

To use the REST APIs, you must be an authorized Illumio user and have credentials to log into the PCE.

For authentication permissions for each REST API call, see the [Illumio Core REST API Reference](#).

User Permissions and the API

Authentication to the PCE is based on three user roles that allow users to perform specific API operations:

- Organization owner: All GET, POST, PUT, and DELETE APIs
- Administrator: Most GET, POST, PUT, and DELETE APIs
- Read-only: GET only

The PCE also has two other kinds of roles:

- Unscoped: Not bound by label scopes
- Scoped: Bound by label scopes

Unscoped Roles

API Role Name	UI Role Name	Granted Access
owner	Global Organization Owner	Perform all actions: Add, edit, or delete any resource, organization setting, or user account.
admin	Global Administrator	Perform all actions except cannot change organization setting and cannot perform user management tasks.
read_only	Global Read Only	View any resource or organization setting. Cannot perform any operations.
global_object_provisioner	Global Policy Object Provisioner	Provision rules containing IP lists, services, and label groups, and manage security settings. Cannot provision rulesets, virtual services, or virtual servers, or add, modify, or delete existing policy items.

Scoped Roles

API Role Name	UI Role Name	Granted Access
ruleset_manager	Full Ruleset Manager	Add, edit, and delete all rulesets within the specified scope. Add, edit, and delete rules when the provider matches the specified scope. The rule consumer can match any scope.
limited_ruleset_manager	Limited Ruleset Manager	Add, edit, and delete all rulesets within the specified scope. Add, edit, and delete rules when the provider and con-

API Role Name	UI Role Name	Granted Access
		sumer match the specified scope. Ruleset Managers with limited privileges cannot manage rules that use IP lists, user groups, label groups, or iptables rules as consumers, or rules that allow internet connectivity.
ruleset_provisioner	Ruleset Provisioner	Provision rulesets within a specified scope. This role cannot provision virtual servers, virtual services, SecureConnect gateways, security settings, IP list, services, or label groups

Session Credentials

While API keys provide a persistent means of authenticating with the PCE, session credentials provide a temporary means of authenticating so you can make Illumio REST API calls.

Choosing to use a session token or an API key depends on your programming needs.

About Session Credentials and Tokens

When you create session credentials, an `auth_username` and `session_token` are returned that function as a temporary username and password for making API calls.

Session credentials expire after not being used for 30 minutes and reset for another 30 minutes if used within the 30-minute window. Session credentials can be used to make all Illumio REST API calls that require authentication, composed of an `auth_username` and a token.

The session token expires after 10 minutes of inactivity.

When to Use a Session Token

An `auth_username` and session token are useful for one-time uses of the API as well as for testing the API.

For example, to write a script that performs a one-time use of the API with a session token, use the Login API to create the `auth_username` and session token. Use those credentials for making other API calls in the script, and then once the script has run, the session token immediately expires when the user logs out..

What Does a Session Token Look Like?

When you authenticate with the PCE using the Login API, the response returns the credentials needed to make other API calls:

- Your username: "auth_username": user_3
- Your session token: "session_token": "xxxxxxx563199f92af7b705ddca26854205b5233"

To use the Illumio REST API:

1. Call `login_users/authenticate` using the e-mail address and password you used to create your PCE account to obtain an *authentication token*.



NOTE:

The authorization token expires after 30 seconds, so have the next call formed and ready to paste onto the terminal window before calling `login_users/authenticate`.

2. Call `users/login` with the authentication token to obtain temporary session credentials.

Authenticate to Login Service

Before you can use the Illumio REST API to access the PCE, you need to use the Login Users API to authenticate with the Illumio Login Service and obtain an authentication token. This authentication token expires in 30 seconds.

The URL for the Illumio Login Service for Illumio Core Cloud users is:

- Login Server: `https://login.illum.io:443`
- PCE: `scp1.illum.io`

If you have deployed the PCE as software, then the hostname for the PCE is the value you defined for the `'pce_fqdn'` parameter in the `runtime_env.yml` file.

Once obtained, you can then pass the authentication token to the PCE you want to access using the Login API. Once you have authenticated with the PCE and obtained a session token, you can make other API calls or [Create an API Key](#) for persistent API access to the PCE.

URI to Authenticate with the Login Service

```
POST [api_version]/login_users/authenticate
```

Parameters

To create an authentication token and authenticate with the Login Service, you need to specify the Fully Qualified Domain Name (FQDN) of the PCE you want to access in the call.

Parameter	Description	Type	Required
pce_fqdn	Fully Qualified Domain Name (FQDN) of the PCE If you are using Illumio Core Cloud, then the FQDN for the PCE is <code>scp1.illum.io</code> . If you have deployed the PCE virtual appliance in your own network, then use the FQDN specified during installation of the PCE virtual appliance.	String	Yes

Curl Command for Authentication

When you received your invitation, you used an e-mail and password to create your PCE account. Use these credentials now to make a call to authenticate.

If you haven't received an invitation, contact your Illumio administrator.

Example (local users only, use SAML ID for remote users):

- `joe_user@example.com` (username)
- `password` (password)

You also need the FQDN of the Login Server plus the FQDN of the PCE host you want to access:

- The Login Server FQDN for Illumio Core Cloud users is `https://login.illum.io:443`
- The PCE FQDN is `scp1.illum.io`

Use this curl command to authenticate with the Illumio Login Service:



NOTE:

The authorization token that is returned (`auth_token`) expires after being idle for 30 seconds, so be ready to call `GET users/login` to create session credentials immediately after making the call to `login_users/authenticate`.

Curl Commands to Retrieve Token

This curl example shows how SaaS local users can use the Illumio Login Service (SAML ID for Remote Users)

```
curl -i -X POST https://login.illum.io:443/api/v2/login_users/authenticate?pce_fqdn=scp1.illum.io -u joe_user@example.com:'password' -H "Content-Type: application/json"
```

Illumio on-premises solutions do not use a login server, so the curl command will look like this:

```
curl -i -X POST -u joe_user@my-company.com:password https://pce.my-company.com:8443/api/v2/login_users/authenticate?pce_fqdn=pce.my-company.com -H "Content-Type: application/json"
```

Response Body to Authenticate with Login Service

The response for the Login Users API is an authentication token (in blue font):

```
{ "auth_token": "xxxxxxxxxxxxxxxxxxxxxxxxw89QutJ5wLntqz5jUrI2guA1rZJXKfcbwUF" }
```

Create Session Credentials Using Login API

Unless you're using persistent API credentials, every time you want to access the Illumio REST API, you must authenticate with the PCE using an *auth username* and a *session token*. To create these session credentials, call GET `/users/login` with the authentication token previously returned by a call to POST `/login_users/authenticate`.

URI

```
GET [api_version]/users/login
```

Parameters

Login Service authentication token you obtained using the Login Users API.

Login Users API JSON Schema

This API uses the Illumio Core schema `users_login_get.schema.json`.

Curl Command to Create Session Token

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/users/login -H "Authorization: Token token=ntqz5jUrI2guA1XzUiLCJlBmMi0iJBMTI4Q0JDLUhZJ"
```

Response Body

GET /users/login returns a temporary `auth_username` and `session_token` shown below in blue. These are used in the curl examples as `$KEY:$TOKEN` respectively (if you're not using persistent API credentials).

Example: `-u user_4:'xxxxxxxx628f5773c47b72dbcd437b4a10d85a06a'`

```
{
  "full_name": "Buford T. Justice",
  "local": true,
  "type": "local",
  "href": "/users/4",
  "auth_username": "user_4",
  "inactivity_expiration_minutes": 10,
  "start": "2017-10-12 16:49:49 UTC",
  "time_zone": "America/Los_Angeles",
  "last_login_ip_address": "209.37.96.18",
  "last_login_on": "2017-10-12T16:49:49.000Z",
  "certificate": {
    "expiration": "2017-11-27T03:09:00.000Z",
    "generated": false
  },
  "login_url": "https://devtest166.ilabs.io:8443/login",
  "orgs": [
    {
      "org_id": 1,
      "org_href": "/orgs/1",
      "display_name": "illum.io",
      "role_scopes": [
        {
          "role": {
            "href": "/orgs/1/roles/owner"
          },
          "scope": [],
          "href": "/orgs/1/users/4/role_scopes/4"
        }
      ]
    }
  ],
  "session_token": "xxxxxxxx628f5773c47b72dbcd437b4a10d85a0",
```

```
"version_tag": "60.1.0-9701f78bef46f521e3d6dd98f70cd8c220940885",
"version_date": "Tue Sep 12 11:12:46 2017 -0700",
"product_version": {
  "version": "17.1.1",
  "build": "6168",
  "long_display": "17.1.1-6168",
  "short_display": "17.1.1"
}
}
```

Curl API Call Using Session Credentials

Once you obtain an `auth_username` and session token from the PCE, you use them to make API calls.

For example, if you wanted to use this session token to get a collection of labels in an organization using the [Labels API](#), the curl command can be written as shown below, using the following authentication:

- `auth_username: user_3`
- `Session Token: xxxxxx563199f92af7b705ddca26854205b5233`

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/labels -H "Accept: application/json" -u user4:'xxxxxxxx628f5773c47b72dbcd437b4a10d85'
```

API Keys

This Public Stable API allows you to create API keys you can use to make API calls to the PCE. Using an API key provides a persistent means of authenticating with the PCE and is recommended for scripting.

In addition to creating an API key, you can also get an individual or collection of API keys created by your user, update an API key, and delete an API key.

About API Keys

API credentials (keys) are persistent and do not time out. API credentials are composed of an `auth_username` and an API secret.

Optionally, you can call `api_keys` with your session credentials to obtain persistent API credentials.

When you create an API key, you receive an `api_username` and `secret`, which function as the username and password for making API calls. An API key is permanent and does not expire (unless when deleted).

When to Use an API Key

Use API keys to write scripts that run automatically, without requiring a human user to authenticate the API call. Unless you are a read-only user, you can create multiple API keys and make API calls in your scripts.

You can also create different API keys for different functions. For example, you might use an API key for scripting automatic workload pairing, and another API key for collecting system events from Illumio.



NOTE:

Before you can create an API key using the Illumio REST API, use the Login API to create a user ID and session token.

What Does API Key Look Like?

When you create an API key, the response returns both the `auth_username` and the `secret` needed for authenticating other API calls:

- API username: `"auth_username": "api_XXXXXXXXXX29"` (represented in the code examples in this document as `$KEY`)
- API key secret: `"secret": "xxxxxxx5048a6a85ce846a706e134e-f1d4bf2ac1f253b84c1bf8df6b83c70d95"` (represented in the code examples in this document as `$TOKEN`)

API Key Methods

Functionality	HTTP	URI
Get a collection of API keys	GET	<code>[api_version][user_href]/api_keys</code>
Get an individual API key	GET	<code>[api_version][api_key_href]</code>
Create an API key	POST	<code>[api_version][user_href]/api_keys</code>
Update an API key	PUT	<code>[api_version][api_key_href]</code>
Delete an API key	DELETE	<code>[api_version][api_key_href]</code>

Get API Keys

When you GET an individual API key or a collection of API keys, the response only returns those API keys created by the user that has authenticated with the PCE for the

session.

This API gets one API key or a collection of API keys that a specific user has created. To get a single API key, you need to know the API key's URI, which is returned in the form of an HREF path when you create an API key, as well as the HREF of the user who created the key.

URI to Get an Individual API Key

```
GET [api_version][api_key_href]
```

URI to Get a Collection of API Keys

```
GET [api_version][user_href]/api_keys
```

Query Parameters

Property	Description	Type
user_id	The user ID in the form of an HREF (e.g., 'users/6') of the user who created the API key.	String
key_id	This is the actual API key ID. Use this query parameter only for a GET instance call.	String

Response Properties

Property	Description	Type	Required
key_id	The unique ID of the API key.	Integer	Yes
auth_user-name	Username associated with the API key required for authentication.	String	Yes
created_at	Timestamp when this key was first created (RFC 3339) in date-time format.	String	Yes
name	The unique name to give the key. Can be any string.	String	Yes
description	The description of the key.	String	No
href	URI of the API key.	String	Yes

Curl Command to Get a Key

The API key is identified in the form of an HREF path property:

```
"/users/11/api_keys/a034248fbcdd60b4"
```

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/users/11/api_
keys/a034248fbcdd60b4 -H "Accept: application/json" -u $KEY:$TOKEN
```

Curl Command to Get Collection of Keys



IMPORTANT:

To use an API key, store the key and secret safely. Anyone with access to both has access to your organization's API.

Due to security concerns, external users are not allowed to create an API Key even if their roles allow it.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/users/11/api_keys -H
"Accept: application/json" -u $KEY:$TOKEN
```

Response Body

An API key is represented by its HREF path, as shown here:

```
/users/29/api_keys/1e9bb1787883639d5
```

For example:

```
[
  {
    "href": "/users/29/api_keys/1e9bb1787883639d5",
    "key_id": "1e9bb1787883639d5",
    "auth_username": "api_1e9bb1787883639d5",
    "created_at": "2016-01-27T01:30:22.274Z",
    "name": "my_api_key",
    "description": "my_scripting_key"
  },
  {
    "href": "/users/29/api_keys/1793df73a99255f7e",
    "key_id": "1793df73a99255f7e",
    "auth_username": "api_1793df73a99255f7e",
    "created_at": "2016-03-14T16:20:43.603Z",
    "name": "MyKey",
    "description": "My Special Key"
  }
]
```

```
}  
]
```

Curl Command to Get All Labels with Key

If you use an API key to get a collection of labels in an organization, and your API key uses these credentials:

- `api_xxxxxxx64fcee809` is the API key
- `xxxxxxx09137412532289d6ecd10bc89c6b1f608c9a85482e7a573` is the secret (API key password)

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/labels -H "Accept:  
application/json" -u api_  
xxxxxxx64fcee809:'xxxxxxx09137412532289d6ecd10bc89c6b1f608c9a85482e7a573'
```

Session and persistent (API key) credentials are represented in this document as the constants `$KEY:$TOKEN` (with no spaces).

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/labels -H "Accept:  
application/json" -u $KEY:$TOKEN
```

Create an API Key

This API creates a unique API key and returns an API key ID and secret, which you can use to get, update, or delete the key, and to make other API calls.

To create an API key, you first need to authenticate either using a session token or another API key. To obtain a session token, use the [Users API](#) and authenticate with the PCE. You will receive your user ID, user HREF, and a session token that you can use when you call this API to create an API key.



IMPORTANT:

If you use an API key, safely store the key and the secret. Anyone with access to both will have access to the API for your organization.



IMPORTANT:

Due to security concerns, external users are not allowed to create an API Key even if their roles allow it.

URI

```
POST [api_version][user_href]/api_keys
```

An example user HREF looks like this:

```
/users/99
```

Request Body

Property	Description	Type	Required
name	The unique name to give the key. Can be any string.	String	Yes
description	The description of the key.	String	No

Payload to Create an API Key

```
{
  "name": "my_api_key",
  "description": "my_scripting_key"
}
```

Curl Command to Create an API Key

In this curl command, the user authentication (-u) uses the session credentials returned from calling the Login API to log in a user. The API key is passed as a JSON object formatted inside of double quotes in the command:

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/users/14/api_keys -H
"Content-Type:application/json" -u user_14:'xxxxxxx563199f92af7b705ddca2685' -d "{
"name":"my_api_key","description":"my_scripting_key" }"
```

Response Body

This example shows the response from creating an API key, which you can use for making other API calls. These values do not expire. The auth_username functions as the username, and the secret functions as the password when making other API calls:

```
{
  key_id: "xxxxxxx6654188229"
  secret: "xxxxxxxxxxa6a85ce846a706e134ef1d4bf2ac1f253b84c1bf8df6b83c70d95"
```

```
auth_username: api_XXXXXX6654188229
}
```

These values can now be used to authenticate with the API as follows:

- Username: `api_XXXXXX29api_XXXXXX6654188229`
- Password: `XXXXXXXXXXa6a85ce846a706e134ef1d4bf2ac1f253b84c1bf8df6b83c70d95`

Create a Key Using the PCE Web Console

You can also create API keys in the PCE web console with the **User** Menu.

1. In the drop-down **User** menu, select **My API Keys**.
A list of configured API keys is displayed.
If no API keys are configured, the message "No API Keys" is displayed.
2. To add a new API key, click **Add**.
3. In the *Create API Key* pop-up window, enter a name for the API key in the Name field. Optionally, enter a description in the Description field.
4. Click **Save** to save your API key or click **Cancel** to close the pop-up window without saving your changes.
5. When the *API Key Created* window appears, click the > button next to "Show credentials" to display the credentials for your API key.

The following information is displayed:

- *Key ID*: The unique ID of the API key
 - *Authentication Username*: The username that authenticates the API calls
 - *Secret*: The password for the API key
6. Click **Download Credentials** to download the credentials as a text file. Make sure that you have saved the credential information before clicking **Done**.

After you click **Done**, the API Keys page displays a summary of your new API key, including the following information:

- Name
- Description
- Key ID
- Authentication Username
- Created On

**NOTE:**

The credential information is displayed only once. Make sure to save it in a secure location because it is used to access the API for your organization. If the credential information is lost, you must create a new API key.

Update an API Key

This API allows you to update an API key name or description. To make this call, you need the API key URI, which is returned in the form of an HREF path when you [Create an API Key](#).

URI to Update an API Key

```
PUT [api_version][api_key_href]
```

Request Body

Property	Description	Type	Required
name	The unique name to give the key. Can be any string.	String	Yes
description	The description of the key.	String	No

Example Payload

```
{
  "name": "my_api_key1",
  "description": "my_scripting_key v2"
}
```

Curl Command to Update an API Key

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/users/99/api_
keys/a034248fbcdd60b4 -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{
"name": "my_key_1", "description": "my_scripting_key v2" }'
```

Delete an API Key

To delete an API key, you need the unique API key ID, which is returned in the form of an HREF path property when you either create a new API key, or when you get a single or a collection of API keys.

URI to Delete an API Key

```
DELETE [api_version][api_key_href]
```

Curl Command to Delete an API Key

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/users/11/api_
keys/043902c883d133fa -u api_
xxxxxxx73752bf29: 'xxxxxx5048a6a85ce846a706e134ef1d4bf2ac1f253b84c1bf8df6b83c70d95'
```

REST API Users

This Public Stable API allows you to log your User into the PCE so you can get a session token to access other Illumio Core REST API calls. This API is your starting point for interacting with the PCE using the REST API.

Users API Methods

Functionality	HTTP	URI
Authenticate to the Illumio Login Service and obtain a single-use authentication token.	POST	[api_version]/login_users/authenticate
Create a new user.	POST	[api_version][users]
Log in a user and obtain a session token.	GET	[api_version]/users/login
Log out a user and destroy the session token.	PUT	[api_version][user_href]/logout
Get a user's information.	GET	[api_version][user_href]
Update user's information.	PUT	[api_version][user_href]
Change a user's password (a local, non-SSO user).	PUT	[api_version]/login_users/[user_href]/password

Log Into the PCE

URI to Log In User

```
GET [api_version]/users/login
```


For step-by-step instructions about how to authenticate to the PCE and use `GET /users/login` in conjunction with other methods, see [Authentication and API User Permissions](#).

Log Out and Destroy Session Credentials

This API logs users out of the PCE and destroys the temporary session credentials used to log them in.



NOTE:

This `PUT /logout` call is not used with persistent API credentials.

URI to Log Out a User

```
PUT [user_href]/logout
```

Request Body

The request body is an empty JSON object.

```
{}
```

Curl Command to Log Out a User

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/authentication_
services/password_policy -H "Content-Type: application/json" -u $KEY:$TOKEN -d '
{"require_type_symbol": true, "expire_time_days": 90}
```

Get User Information

This API gets specific information about a user, such as when a user logged into the Illumio PCE, the IP address from where the user logged in, the user's name, and password.

URI to Get User Information

```
GET [user_href]
```

Response Properties

Property	Description	Type
href	URI of the user.	String

Property	Description	Type
username	Username used for authentication.	String
last_login_on	When the user logged on.	String
last_login_ip_address	The IP address of the system where the user has logged into the PCE.	String
login_count	The number of times the user has logged in.	Integer
full_name	Full name of a user as listed in the PCE web console.	String
time_zone	User's timezone IANA Region name.	String
locked	Indicates if a user account is locked or not. True = locked.	Boolean
effective_groups	A list of group names to which the user belongs.	String
local_profile	Local user profile	Object
updated_at	Date when user account information was last updated in the system.	String
created_at	Date when the user account was created in the system.	String
type	Indicates if the user account is authenticated by the PCE (local) or by a third party SAML-based identity management system (external)	String

Request Example

```
GET https://pce.my-company.com:8443/api/v2/users/5
```

Curl Command to Get a User's Information

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/users/14 -H "Accept: application/json" -u $KEY:$TOKEN
```

Response Body

In this response, the user is represented in the system by an HREF path property ("href": "/users/14") that can be used when you want to update the user information.

```
{
  "href": "/users/14",
  "type": "local",
  "effective_groups": [],
```

```
"id": 14,
"username": "joe.user@pce.my-company.com",
"full_name": "Ralph W. Emerson",
"time_zone": "America/Los_Angeles",
"locked": false,
"login_count": 75,
"last_login_ip_address": "xxx.37.96.18",
"last_login_on": "2017-08-17T15:42:25.732Z",
"local_profile": {
  "pending_invitation": false
},
"created_at": "2015-10-26T05:24:08.735Z",
"updated_at": "2017-08-17T15:55:40.130Z"
}
```

Create a New User

This API creates a new local user.

URI to Create a New User

```
POST [api_version][users]
```

Request Body

Property	Description	Type	Required
full_name	User's full name.	String	No
username	username is an e-mail address such as user@example.com	String	Yes
type	User's type, such as user authenticated as local.	String	Yes
time_zone	The user's timezone IANA region name.	String	No

Curl Command to Create a User

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/users/users
```

Possible Responses

When you execute the command to update a user, you can receive one of these three messages:

- 204 success: A new local user was created successfully.
- 406: Validation errors such as `invalid`.
- 501: The user is created, but the invitation e-mail failed. The new user cannot register or sign-up. If you receive this message, you need to create another local user.

Resend Invitation for a Local User

To resend the invitation to a new local user after an e-mail notification failure, use the following URI:

```
PUT /users/:user_id/local_profile/reinvite
```

Update User Information

This API updates an Illumio API user's account information.

URI to Update User's Information

```
PUT [api_version][user_href]
```

Request Body

The request body is an empty JSON object.

```
{}
```

If you attempt to use a PUT with that URL without a payload, the 406 error shows `No payload provided for PUT request`.

Property	Description	Type	Optional
<code>full_name</code>	User's full name	String	Yes
<code>time_zone</code>	The user's time zone IANA region name	String	Yes

Example Payload to Update an API User's Information

```
{  
  "name": "Billy T. Kidd"  
}
```

Curl Command to Update User's Information

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/users/11 -H "Content-Type: application/json" -u $KEY:$TOKEN -d '{"name": "Billy T. Kidd"}'
```

Change the User Password

This API method allows currently authenticated users to change their login password.

- The call must be made **by the user currently authenticated** in the session; even an administrator cannot change another user's password.
- An API key is not used with this API.
- The user's login name (typically the user's e-mail address) and login password are used for authentication.
- The user's five most recent passwords cannot be used.

URI to Change the User's Password

```
PUT [api_version]/login_users/[user_href]/password
```

Request Body

Property	Description	Type	Required
password	User's new password must meet these requirements: <ul style="list-style-type: none">• Have a minimum of 8 characters• Have at least 1 capital letter• Have at least 1 lowercase letter• Have at least 1 number• Not match previously used passwords	String	Yes

Example Request Body to Change an API User's Password

```
{
  "password": "'new_password'"
}
```

Curl Command to Change the User's Password

```
curl -u 'username'@'company'.com:'existing_password' -X PUT
https://'company'.com:8443/api/v2/login_users/me/password -H "Content-type:
application/json" -d '{"password":"'new_password'"}' -i
```

Possible Responses

When you execute the command to change a password, you can receive one of these three messages:

- 204 success: The password was changed successfully.
- 406: Validation error such as `invalid`.
- 501: Password is changed, but e-mail notification failed.

LDAP Authentication

The new Public Experimental API provides user authentication with the PCE using LDAP with OpenLDAP and Active Directory.

LDAP authentication comes in addition to the two previously available methods:

- API keys, which provide persistent authentication, and
- Session credentials, which provide temporary authentication.

Prerequisites and Limitations

Before configuring LDAP for authentication with the PCE, it is important to provide the required prerequisites and review any limitations.

Determine Your User Base DN

Before you map your LDAP settings to PCE settings, determine your user base Distinguished Name (DN). The DN is the location in the directory where authentication information is stored.

If you don't have this information, contact your LDAP administrator for assistance.

When configuring the PCE to work with LDAP, be aware of the following:

- PCE uses LDAP protocol version 3 ("v3").
- Supported LDAP distributions include OpenLDAP 2.4 and Active Directory.
- Supported LDAP protocols include LDAP, LDAPS, or LDAP with STARTTLS.

Limitations

These are the current limitations for LDAP authentication:

- Any locally created user has precedence over an LDAP user of the same name. For example, if the LDAP server has a user with a username attribute (such as `cn` or `uid`) of *johndoe* and the default PCE user of the same name is present, the PCE user takes precedence. Only the local password is accepted. On login, the roles mapped to the local user will be in effect. To work around this limitation, you must delete the specific local user.
- LDAP and SAML single sign-on authentication methods cannot be used together. In this release of the PCE, an organization can either use LDAP or SAML single sign-on for authenticating external users.
- This release enables LDAP configuration via REST APIs only.

Configuring LDAP Authentication for the PCE

The PCE supports user and role configuration for LDAP users and groups. You can configure up to three LDAP servers and map users and user groups from your LDAP servers to PCE roles.

For information about configuring multiple LDAP servers, see [How the PCE Works with Multiple LDAP Servers](#).

Before you configure LDAP, review the LDAP prerequisites and considerations topic in this document.

Authentication Precedence

PCE local authentication takes precedence over any external systems. The PCE authenticates a user in the following order:

- a. The PCE first attempts local authentication. If the account is expired or otherwise fails, the PCE does not try to log in by using LDAP authentication.
- b. If the local user does not exist, the PCE attempts LDAP login (if enabled).

Configuration Steps

To configure the PCE to work with LDAP, perform these steps:

1. Enable the PCE to use LDAP authentication. See [Enable LDAP Authentication](#).
2. Set up an LDAP configuration. See [Configure LDAP Authentication](#).

When searching for LDAP users, the PCE follows the order in which the servers were configured. The configurable request timeout is 5 seconds by default. Once the request time expires, the PCE attempts to connect to the next server in the configuration.

For example, assume that you configure three LDAP servers in this order: A, B, and C. The PCE will search the servers in that same order. If it finds a user on server A, it stops even if the same user also exists on servers B and C. The PCE will try to use A's credentials for that user, but if it fails to connect to A, it searches the remaining servers: first B, then C. The search proceeds following the expiration of the connection timeout.

3. Map your LDAP groups to one or more PCE roles. See [Mapping LDAP Group Membership to PCE User Roles](#).

Mapping LDAP Group Membership to PCE User Roles

First, configure the PCE to use LDAP authentication. Second, map PCE roles to that server's groups.

When a user attempts to log in, the PCE queries the server(s) to find that user. It grants the user permissions based on any roles associated with the LDAP groups to which the user belongs.

You have the following options for changing user permissions:

- For a group of users, remap the LDAP group to a different PCE role.
- For an individual user, move the user to an LDAP group mapped to a different PCE role using the LDAP server.

You can also perform these user management activities:

- Add a user to a PCE role:
 - On the PCE, map the PCE role to an LDAP group.
 - On your LDAP server, add the user to that LDAP group.
- Remove a user from a PCE role by removing it from the corresponding LDAP group on your LDAP server.

Users can have memberships in several roles. In that case, they have access to all the capabilities available for any of these roles. For example, a user is a member of both the **docs** and **eng** groups and **docs** group is mapped to "Ruleset Manager" while the **eng** group is mapped to "Ruleset Provisioner." In this case, the user obtains all permissions assigned both to the "Ruleset Manager" and "Ruleset Provisioner" roles.



NOTE:The PCE checks LDAP membership information when a user attempts to log in. You do not need to reload the authentication configuration when adding or removing users.

See the PCE Web Console Guide for information about the mapping from external groups to PCE user roles.

How to set up the PCE for LDAP Authentication

The PCE supports LDAPS and LDAP with STARTTLS. To use the PCE with secure LDAP with SSL/TLS certificates, add the certificate chain to the local certificate store on the PCE.

Using REST APIs for LDAP Configuration in the PCE

The following table provides an overview of the REST APIs you have available to configure the PCE for LDAP Authentication. For information about the parameters for these REST APIs, see [LDAP Configuration Parameters](#) and [REST API Schema Files](#).

APIs for LDAP Configuration

PCE APIs	HTTP	URI
Retrieve the PCE authentication settings	GET	[api_version]/authentication_settings
Update the PCE authentication settings	PUT	[api_version]/authentication_settings
Retrieve the LDAP configuration	GET	[api_version]/authentication_settings/ldap_configs
Get instance	GET	[api_version]/authentication_settings/ldap_configs/:uuid
Create an LDAP configuration	POST	[api_version]/authentication_settings/ldap_configs
Update an LDAP configuration	PUT	[api_version]/authentication_settings/ldap_configs/:uuid
Delete an LDAP configuration	DELETE	[api_version]/authentication_settings/ldap_configs/:uuid
Verify the connection to the LDAP server	POST	[api_version]/authentication_settings/ldap_configs/:uuid/verify_connection

LDAP Configuration Parameters

API Property Name	Type	Required	Description
pce_fqdn	String	No	Regional PCE member FQDN for Super-cluster. For non-supercluster deployment, it is the FQDN of the PCE cluster.
name	String	No	Friendly name of the LDAP server
address	String. Format: hostname or ipv4	Yes	IP address or hostname of the LDAP server
port	Integer	Yes	Port number of the LDAP server - 636 for LDAPS or 389 for STARTTLS
authentication_method	Enum	Yes	<ul style="list-style-type: none"> LDAP: Clear text connection LDAPS: LDAP over SSL/TLS Protocol STARTTLS: LDAP over SSL/TLS Protocol with handshake establishment before Secure connection:
request_timeout_seconds	Integer	No	Number of seconds to wait for a response; default 5 seconds. Possible values: 1-60
bind_distinguished_name	String	Yes	Distinguished name (DN) used to bind to the LDAP server
bind_password	String	No	Bind DN password. Only applicable for POST or PUT operations; attribute will not be returned for GET instance or collection APIs.
is_bind_password_set	boolean	No	Flag to indicate whether Bind DN password is configured. Adding this flag because the API does not return the bind password and there is a need to indicate if the password has been set for the bind_distinguished_name. Only applicable for GET operation
user_base_distinguished_name	String	Yes	Base DN to search for users

API Property Name	Type	Required	Description
user_distinguished_name_pattern	String	No	Pattern used to create a DN string for a user during login; for example, uid=*,ou=people, where * will be replaced with the username
user_base_filter	String	No	Search filter used to query the LDAP tree for users
username_attribute	String	Yes	Attribute on a user object that contains the username; for example, uid, sAMAccountName, userPrincipalName
full_name_attribute	String	No	Attribute on a user object that contains the full name; for example, cn, commonName, displayName
user_memberof_attribute	String	No	Attribute on a user object containing group membership information; for example, memberOf, isMemberOf
insecure_disable_tls_certificate_verification	boolean	No	Specifies whether to verify the server certificate when establishing an SSL connection to the LDAP server; default false

Enable LDAP Authentication

This section explains how to use API to enable the PCE for LDAP authentication. You must enable the LDAP preview feature in the PCE before invoking this API. For the steps to enable this preview feature, see [Enabling the LDAP Authentication Preview](#).

URI

PUT /api/v2/authentication_settings

Request Body

Property	Data Type	Required	Description
authentication_type	enum	Yes	The type of authentication

Enum Item	Purpose
Local	Local DB authentication

Enum Item	Purpose
SAML	SAML authentication enabled
RADIUS	RADIUS authentication enabled
LDAP	LDAP authentication enabled

Example Payload to Configure LDAP Authentication

```
{
  "authentication_type": "LDAP",
}
```

Response Code

The following response codes can be returned:

- 200 indicates success
- 403 indicates the user is not an org owner
- 406 indicates invalid parameters

Configure LDAP Authentication

This API creates the configuration for an LDAP server in the PCE. For information about the request parameters, see [LDAP Configuration Parameters](#).

URI

POST /api/v2/authentication_settings/ldap_configs

Request body for a multi-node cluster

```
{
  "name" : "ldap 1" ,
  "address" : "ldap-1.mycompany.com " ,
  "port" : "10636" ,
  "authentication_method" : "LDAPS" ,
  "request_timeout_seconds" : 4,
  "bind_distinguished_name" : 'CN=admin,CN=Users,DC=mycompany,DC=com' ,
  "bind_password" : 'test1234' ,
  "user_base_distinguished_name" : 'DC=mycompany,DC=com' ,
  "username_attribute" : 'sAMAccountName' ,
  "full_name_attribute" : 'cn' ,
}
```

```
"user_memberof_attribute" : 'memberof',  
}
```

Request body for a supercluster

```
{  
  "pce_fqdn" : "devmr01" ,  
  "name" : "ldap 1" ,  
  "address" : "ldap-1.mycompany.com" ,  
  "port" : "10636" ,  
  "authentication_method" : "LDAPS" ,  
  "request_timeout_seconds" : 4,  
  "bind_distinguished_name" : 'CN=admin,CN=Users,DC=mycompany,DC=com' ,  
  "bind_password" : 'test1234' ,  
  "user_base_distinguished_name" : 'DC=mycompany,DC=com' ,  
  "username_attribute" : 'sAMAccountName' ,  
  "full_name_attribute" : 'cn' ,  
  "user_memberof_attribute" : 'memberof' ,  
}
```

Response Code

The following response codes can be returned:

- 204 indicates success
- 403 indicates the user is not an org owner
- 406 indicates invalid parameters

Configure Secure LDAP

In the process of configuring an LDAP server in the PCE, you need to configure LDAP for SSL authentication.

You can Secure LDAP with SSL/TLS Certificates using these three methods:

- Use PCE Web UI to Configure Secure LDAP.
- Install LDAP TLS Certificates to the PCE System CA Store from the PCE Command-Line.
- [Configure LDAP for SSL authentication](#) using REST APIs

Configure LDAP for SSL authentication

The following APIs are used to configure LDAP for SSL:

- GET /authentication_settings/ldap_configs
- GET /authentication_settings/ldap_configs/:uuid
- POST /authentication_settings/ldap_configs
- PUT /authentication_settings/ldap_configs/:uuid

The required property is `tls_ca_bundle`.

To manage TLS CA bundle for LDAP authentication use these APIs:

- GET /login_proxy_ldap_configs
- POST /login_proxy_ldap_configs
- PUT /login_proxy_ldap_configs/update

Update LDAP configuration

This section outlines how to update the LDAP server configuration in the PCE. For information about the request parameters, see [LDAP Configuration Parameters](#).

URI

PUT /api/v2/authentication_settings/ldap_configs/:uuid

(uuid indicates the LDAP server configuration uui)

Request Body

```
{
  "address" : "ldap-1.mycompany.com" ,
  "bind_password" : "qw3r!y123!!" ,
  "full_name_attribute" : "displayName" ,
  "port" : 636,
  "insecure_disable_tls_certificate_verification": true
}
```

Response Code

The following response codes can be returned:

- 204 indicates success
- 403 indicates the user is not an org owner
- 404 indicates LDAP configuration not found or an attempt to update LDAP configuration in another domain
- 406 indicates invalid parameters

Delete LDAP Server Configuration

This API deletes the configuration for an LDAP server in the PCE. For information about the request parameters, see LDAP Configuration Parameters Overview.

URI

DELETE /api/v2/authentication_settings/ldap_configs/:uuid

uuid indicates the LDAP server configuration uuid

Request Body

None

Response Code

The following response codes can be returned:

- 204 indicates success
- 403 indicates the user is not an org owner
- 404 indicates LDAP configuration not found or an attempt to update LDAP configuration in another domain
- 406 indicates invalid parameters

Test LDAP Server Connectivity

This section outlines the use of the API to verify the connectivity for a configured LDAP server in the PCE.

URI

POST /api/v2/authentication_settings/ldap_configs/:uuid/verify_connection

(uuid indicates the LDAP server configuration uuid)

Request Body

none

Response Body

If a server connection is verified successfully:

```
{
  "verified" : true
}
```

If the server connection verification fails:

```
{
  "verified" : false ,
  "errors" : [
    {
      "token" : "ldap_server_verification_failure" ,
      "message" : "LDAP server verification failure: LDAP server error
message"
    }
  ]
}
```

Response Code

The following response codes can be returned:

- 200 indicates success
- 403 indicates the user is not an org owner
- 404 indicates LDAP configuration not found

Use Cases

Configure LDAP for SSL authentication

Use case 1:

Retrieve all LDAP configurations for the domain.

1. Request format: GET /api/v2/authentication_settings/ldap_configs
2. Possible parameters (drawn from REST API conventions):
 - Required: none

- Optional: none
- 3. Request Body: none
- 4. Response format: JSON
- 5. Response Code: 200 success

Use case 2:

Create LDAP server configuration.

1. Request format: POST /api/v2/authentication_settings/ldap_configs
2. Possible parameters (drawn somewhat from REST API Conventions):
 - Required: none
 - Optional: none
3. Request Body:

Single-PCE

```
{
  "name": "ldap 1",
  "address": "ldap-1.ilabs.io",
  "port": "10636",
  "authentication_method": "LDAPS",
  "request_timeout_seconds": 4,
  "bind_distinguished_name":
'CN=admin,CN=Users,DC=ilabs,DC=io',
  "bind_password": 'test1234',
  "user_base_distinguished_name": 'DC=ilabs,DC=io',
  "username_attribute": 'sAMAccountName',
  "full_name_attribute": 'cn',
  "user_memberof_attribute": 'memberof',
  "tls_ca_bundle": "
-----BEGIN CERTIFICATE-----

MIIDhTCCAm2gAwIBAgIQYx+dZzQPBLdN6e8uqW2ByDANBgkqhkiG9w0BAQ0FADBJ
.....
```

```

-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----

MIIF7TCCBNWgAwIBAgITEgAAAEg0ToOKIywtOQAAAAAASDANBgkqhkiG9w0BAQ0F

.....
.....
-----END CERTIFICATE-----"
}

```

Supercluster

```

{
  "pce_fqdn": "devmr01",
  "name": "ldap 1",
  "address": "ldap-1.ilabs.io",
  "port": "10636",
  "authentication_method": "LDAPS",
  "request_timeout_seconds": 4,
  "bind_distinguished_name":
'CN=admin,CN=Users,DC=ilabs,DC=io',
  "bind_password": 'test1234',
  "user_base_distinguished_name": 'DC=ilabs,DC=io',
  "username_attribute": 'sAMAccountName',
  "full_name_attribute": 'cn',
  "user_memberof_attribute": 'memberof',
  "tls_ca_bundle": "-----BEGIN CERTIFICATE-----

MIIDhTCCAm2gAwIBAgIQYx+dZzQPBLdN6e8uqW2ByDANBgkqhkiG9w0BAQ0FADBJ

-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----

MIIF7TCCBNWgAwIBAgITEgAAAEg0ToOKIywtOQAAAAAASDANBgkqhkiG9w0BAQ0F

-----END CERTIFICATE-----"
}

```

4. Response format: JSON

5. Response Code:
 - 204 success
 - 403 not an org owner
 - 406 invalid params

Use case 3:

Update LDAP server configuration:

1. Request format: PUT /api/v2/authentication_settings/ldap_configs/:uuid
2. Possible parameters (drawn somewhat from REST API Conventions):
 - Required: uuid - LDAP server configuration UUID
 - Optional: none
3. Request Body:

```
{
  "tls_ca_bundle": "
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----"
}
```

4. Response format: JSON
5. Response Codes:
 - 204 success
 - 403 not an org owner
 - 404 LDAP config not found or attempt to update LDAP config in another domain
 - 406 invalid params

REST API Schema Files

The following schema files for LDAP configuration are available in 19.3.5:

- ldap_config.schema.json
- authentication_settings_ldap_configs_get.schema.json
- authentication_settings_ldap_configs_post.schema.json
- authentication_settings_ldap_configs_put.schema.json
- authentication_settings_ldap_configs_verify_connection_post.schema.json
- authentication_settings_get.schema.json
- authentication_settings_put.schema.json

Sample Responses

GET /authentication_settings

```
{
  "authentication_type" : "LDAP"
}
```

Single-PCE: GET /authentication_settings/ldap_configs

```
[
  {
    "href": "/authentication_settings/ldap_configs/acf577c8-839a-4828-90f6-797bfc1b54d1",
    "pce_fqdn": "test.ilabs.io",
    "name": "mycompany",
    "address": "ldap-1.mycompany.com",
    "port": 389,
    "authentication_method": "LDAP",
    "request_timeout_seconds": 5,
    "bind_distinguished_name": "john.doe@mycompany.com",
    "is_bind_password_set": true,
    "user_base_distinguished_name": "OU=Users,OU=mycompany
Employees,DC=mycompany,DC=com",
    "user_distinguished_name_pattern": null,
    "user_base_filter": "(&(objectcategory=person)(objectclass=user))",
    "username_attribute": "userPrincipalName",
    "full_name_attribute": "cn",
    "user_memberof_attribute": "memberOf",
  }
]
```

```
"insecure_disable_tls_certificate_verification":false,
"created_at":"2019-03-07T23:30:13.046Z",
"updated_at":"2019-03-07T23:30:13.046Z",
"created_by":{
  "username":"john.doe@mycompany.com"
},
"updated_by":{
  "username":"john.doe@mycompany.com"
}
},
{
  "href":"/authentication_settings/ldap_configs/827b0e34-16ed-4b87-9263-8cc6b9614302",
  "pce_fqdn":"test.ilabs.io",
  "name":"jumpcloud",
  "address":"ldap2.jumpcloud.com",
  "port":636,
  "authentication_method":"LDAPS",
  "request_timeout_seconds":5,
  "bind_distinguished_
name":"uid=test,ou=Users,o=58b6704846cf383825533989,dc=jumpcloud,dc=com",
  "is_bind_password_set":false,
  "user_base_distinguished_
name":"ou=Users,o=58b6704846cf383825533989,dc=jumpcloud,dc=com",
  "user_distinguished_name_
pattern":"ou=Users,o=58b6704846cf383825533989,dc=jumpcloud,dc=com",\
  "user_base_filter":"(objectClass=inetOrgPerson)",
  "username_attribute":"uid",
  "full_name_attribute":"cn",
  "user_memberof_attribute":"memberOf",
  "insecure_disable_tls_certificate_verification":false,
  "created_at":"2019-03-07T23:30:13.046Z",
  "updated_at":"2019-03-07T23:30:13.046Z",
  "created_by":{
    "username":"john.doe@mycompany.com"
  },
  "updated_by":{
    "username":"john.doe@mycompany.com"
  }
}
```

```
    }  
  }  
]
```

Supercluster: GET /authentication_settings/ldap_configs

```
[  
  {  
    "pce_fqdn":"devmr01",  
    "href":"/authentication_settings/ldap_configs/8501dff7-cd3f-4c01-9057-f2b9b1486348",  
    "name":"ldap 1",  
    "address":"ldap-1.mycompany.com",  
    "port":389,  
    "authentication_method":"STARTTLS",  
    "is_bind_password_set":false,  
    "user_base_distinguished_name":"DC=ilabs,DC=io",  
    "user_distinguished_name_pattern":null,  
    "username_attribute":"sAMAccountName",  
    "full_name_attribute":"cn",  
    "user_memberof_attribute":"memberof",  
    "insecure_disable_tls_certificate_verification":false,  
    "created_at":"2018-11-30T18:38:36.634Z",  
    "updated_at":"2018-11-30T18:38:36.634Z",  
    "created_by":{  
      "username":"john.doe@mycompany.com"  
    },  
    "updated_by":{  
      "username":"john.doe@mycompany.com"  
    }  
  },  
  {  
    "pce_fqdn":"devmr01",  
    "href":"/authentication_settings/ldap_configs/827b0e34-16ed-4b87-9263-8cc6b9614302",  
    "name":"ldap 2",  
    "address":"ldap-2.mycompany.com",  
    "port":389,  
    "authentication_method":"STARTTLS",
```

```
"is_bind_password_set":false,
"user_base_distinguished_name":"DC=ilabs,DC=io",
"user_distinguished_name_pattern":null,
"username_attribute":"sAMAccountName",
"full_name_attribute":"cn",
"user_memberof_attribute":"memberof",
"insecure_disable_tls_certificate_verification":false,
"created_at":"2018-12-01T18:38:36.634Z",
"updated_at":"2018-12-01T18:38:36.634Z",
"created_by":{
  "username":"john.doe@mycompany.com"
},
"updated_by":{
  "username":"john.doe@mycompany.com"
}
},
{
  "pce_fqdn":"devmr02",
  "href":"/authentication_settings/ldap_configs/828ca310-d1f9-4125-b88a-
fea8fb0374c5",
  "name":"ldap 1",
  "ldap-1.mycompany.com",
  "port":389,
  "authentication_method":"STARTTLS",
  "user_base_distinguished_name":"DC=ilabs,DC=io",
  "user_distinguished_name_pattern":null,
  "is_bind_password_set":true,
  "username_attribute":"sAMAccountName",
  "full_name_attribute":"cn",
  "user_memberof_attribute":"memberof",
  "insecure_disable_tls_certificate_verification":false,
  "created_at":"2018-12-04T18:38:36.634Z",
  "updated_at":"2018-12-04T18:38:36.634Z",
  "created_by":{
    "username":"john.doe@mycompany.com"
  },
  "updated_by":{
    "username":"john.doe@mycompany.com"
  }
}
```

```
    }
  },
  {
    "pce_fqdn": "devmr02",
    "href": "/authentication_settings/ldap_configs/7fb7e865-1522-4ebd-b614-01bf9180e49d",
    "name": "ldap 2",
    "ldap-2.mycompany.com",
    "port": 389,
    "authentication_method": "STARTTLS",
    "user_base_distinguished_name": "DC=ilabs,DC=io",
    "user_distinguished_name_pattern": null,
    "is_bind_password_set": true,
    "username_attribute": "sAMAccountName",
    "full_name_attribute": "cn",
    "user_memberof_attribute": "memberof",
    "insecure_disable_tls_certificate_verification": false,
    "created_at": "2018-12-04T18:38:36.634Z",
    "updated_at": "2018-12-04T18:38:36.634Z",
    "created_by": {
      "username": "john.doe@mycompany.com"
    },
    "updated_by": {
      "username": "john.doe@mycompany.com"
    }
  }
]
```


Asynchronous GET Collections

This chapter contains the following topics:

Overview of Async GET Requests	65
Async Job Operations	68

When using the standard synchronous GET method on more than the maximum allowed number of 500 resources, only the *latest* 500 results are returned.

To GET all the results when the number of resources exceeds 500, specify in the header that the call is asynchronous (“async”), which then executes the request as an offline job.

Overview of Async GET Requests

An asynchronous job collects all matching records and downloads them as a single job. You can configure a script to continuously poll the job until it is done and then download the results of the job using the job Location HREF listed in the response.

REST Patterns for Collection vs. Instance

GET collection methods return HREF path properties for each individual resource. Perform other REST operations on individual instances of these resources (such as POST, PUT, and DELETE) using the HREF to identify the resources on which to operate.

For example, the response body for the API to get a collection of labels returns a list of labels, where each one is identified as an HREF path. In this instance, the general syntax for the API call looks like this:

```
GET https://scp.illum.io:8443[api_version][org_href]labels
```

[org_href] identifies the organization from which you want to get a collection of labels.

A single label instance in the response is identified by its HREF path:

```
{
  href: "/orgs/2/labels/8"
  key: "env"
  value: "Prod"
  created_at: "2014-01-22T18:24:33Z"
  updated_at: "2014-01-22T18:24:40Z"
  created_by: {
    href: "/users/9"
  }
  updated_by: {
    href: "/users/9"
  }
}
```

To perform other operations on this label (href: ["/orgs/2/labels/8"](/orgs/2/labels/8)), you can provide this HREF in the API call to operate on this label instance.

For example:

```
PUT https://scp.illum.io:8443/api/v2/orgs/2/labels/8
```

Async GET Supported APIs

These APIs support async GET collections:

Description	Resource Type	Exposure
agents/update	GET [api_version][org_href]/agents	Experimental
	GET [api_version][org_href]/agents/update	Experimental
audit_log_events	GET [api_version][org_href]/audit_log_events	Experimental
auth_security_principals	GET [api_version][org_href]/auth_security_principals	Experimental
authentication_settings/ password_policy	GET [api_version][org_href]/authentication_settings/ password_policy	Experimental

Description	Resource Type	Exposure
password_policy		
datafiles	GET [api_version][org_href]/datafiles	Experimental
events	GET [api_version][org_href]/events	Experimental
jobs	GET [api_version][org_href]/jobs	Experimental
labels	GET [api_version][org_href]/labels	Both
network_devices/ network_endpoints	GET [api_version][org_href]/network_ devices/network_endpoints	Experimental
network_enforce- ment_nodes	GET [api_version][org_href]/network_enforce- ment_nodes	Experimental
node_available	GET [api_version][org_href]/node_available	Both
pairing_profiles	GET [api_version][org_href]/pairing_profiles	Experimental
permissions	GET [api_version][org_href]/permissions	Experimental
security_principals	GET [api_version][org_href]/sec_poli- cy/draft/security_principals	Experimental
system_events	GET [api_version][org_href]/system_events	
vulnerability_reports	GET [api_version][org_href]/vulnerability_ reports	Experimental
sec_policy/draft/		
allow	GET [api_version][org_href]/sec_poli- cy/draft/allow	Experimental
dependencies	GET [api_version][org_href]/sec_poli- cy/draft/dependencies	Experimental
ip_lists	GET [api_version][org_href]/sec_poli- cy/draft/ip_lists	Both
label_groups	GET [api_version][org_href]/sec_poli- cy/draft/label_groups	Experimental
label_groups/mem- ber-of	GET [api_version][org_href]/sec_poli- cy/draft/label_groups/member-of	Experimental
modified_objects	GET [api_version][org_href]/sec_poli- cy/draft/modified_objects	Experimental
pending	GET [api_version][org_href]/sec_poli- cy/draft/pending	Experimental
rule_sets	GET [api_version][org_href]/sec_poli- cy/draft/rule_sets	Both
rule_sets/sec_rule	GET [api_version][org_href]/sec_poli- cy/draft/rule_sets/sec_rules	Both

Description	Resource Type	Exposure
services	GET [api_version][org_href]/sec_policy/draft/services	Both
virtual_service	GET [api_version][org_href]/sec_policy/draft/virtaual_services	Both
settings/		
settings	GET [api_version][org_href]/settings	
syslog/destinations	GET [api_version][org_href]/-settings/syslog/destinations	Experimental
workloads	GET [api_version][org_href]/settings/workloads	Experimental
users/		
users	GET [api_version][org_href]/users	Stable
api_keys	GET [api_version][org_href]/users/api_keys	Both
orgs	GET [api_version][org_href]/users/orgs	Experimental
login	GET [api_version][org_href]/users/login	Stable
workloads/		
workloads/	GET [api_version][org_href]/workloads	Both
interfaces	GET [api_version][org_href]/-workloads/interfaces	Both

Async Job Operations

To create the asynchronous GET job request, set the following preference:

```
-H 'Prefer: respond-async'
```

Setting this preference executes the request during low-traffic times as an asynchronous job in the background, which lightens network traffic loads.

Workflow for Async Job Operations

The workflow for requesting an asynchronous bulk job consists of the following tasks:

1. Create the asynchronous GET job request.
2. Poll the job until the status is "Done" or "Failed."
3. Obtain the HREF of the completed request job.
4. Use the HREF to get the results of the request job.

Create an Async Job Request

This example demonstrates a request for an asynchronous collection of labels.



NOTE:

Use query parameters for a filtered job request, such as to return only the environment labels: `.../labels?key=env`

URI to Create a Job Request

```
GET [api_version]/labels
```

The asynchronous collection header is highlighted in **blue bold** font:

```
curl -i -X GET 'https://pce.my-company.com:8443/api/v2/orgs/1/labels' -H 'Accept: application/json' -H 'Prefer: respond-async' -u $KEY:$TOKEN
```

Response with a Job Status

The response is 202 - Accepted, which includes Location, the header Retry-After and an empty body:

```
Server: nginx
Date: Thu, 14 Jan 2016 23:16:52 GMT
"location": https://pce.my-company.com:8443/api/v2/orgs/1/jobs/d1775367-1951-4707-aa2e-37a0b9076d31",
Retry-After: 5
Transfer-Encoding: chunked
Connection: keep-alive
Status: 202 Accepted
Cache-Control: no-cache
X-Request-Id: 36aae8ce-82ed-4a6a-8a76-77d2df78daff
```

Poll the Job

After submitting the job request, poll the job using the suggested Retry-After time to determine when the job is complete.

URI to Get the Status of the Job

The following example demonstrates how to poll the job to determine its status.

```
GET [api_version][org_href]/jobs/[href]
```

Poll the HREF provided in the Location field of the response using the duration specified in Retry-After until the status is either done or failed.

```
curl -i -X GET 'https://pce.my-company.com:8443/api/v2/orgs/1/jobs/[href]' -H
'Accept: application/json' -u $KEY:$TOKEN
```

Async Job Response Properties

The following table defines the properties returned in the response:

Property	Description	Type	Required
href	HREF for resource	String	Yes
job_type	Query type defined during job creation	String	Yes
description	Reference information	String	No
result	Query result	Object (HREF, not required)	Yes
requested_at	Time PCE received request	Date-time	Yes
requested_by	User who initiated request	Object (HREF, required)	Yes
terminated_at	Termination time of job (regardless of outcome)	Date-time	Yes
status	Status of async request	Enum Pending: Waiting to start Running: In progress Done: Complete (successful/unsuccessful) Failed: Unable to complete (exceeded time limit)	Yes
created_by	Creator of request	Object (HREF, required)	Yes

Async Job Status

If the job status is running, the response body includes the following results:

```
{
  "href": "/orgs/1/jobs/43f6e9e3-6a68-4481-87c6-18fd096dafbe",
  "job_type": ":illumio/async_requests",
```

```
"description": "/orgs/1/labels",
"result": {
},
"status": "running",
"requested_at": "2016-01-14 23:16:52.303166",
"requested_by": {
  "href": "/users/1"
}
}
```

Get Async Job Results

The following example demonstrates how to get job results.

URI to Get Async Job Results

```
GET [api_version][org_href]/datafiles/[href]
```

Curl Command to Get Async Job Results

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/1/datafiles/[href] -H
'Accept: application/json' -u $KEY:$TOKEN
```

Response Body with Request Results

When the job is complete, use the HREF in the result field to obtain the results:

```
{
  "href": "/orgs/1/jobs/43f6e9e3-6a68-4481-87c6-18fd096dafbe",
  "job_type": ":illumio/async_requests",
  "description": "/orgs/1/labels",
  "result": {
    "href": "/orgs/1/datafiles/[href]"
  },
  "status": "done",
  "requested_at": "2016-01-14 23:16:52.303166",
  "terminated_at": "2016-01-14 23:17:05.223047",
  "requested_by": {
    "href": "/users/1"
  }
}
```

```
}  
}
```

Poll the Query Job Status

After submitting the job request, poll the job using the suggested "Retry-After" duration to determine when the job is complete.

The PCE has four possible status options for the job:

- Pending: Waiting to start
- Running: In progress
- Done: Complete (successful/unsuccessful)
- Failed: Unable to complete (exceeded time limit)

Get Jobs

Specify the maximum number of jobs to return with the `max_results` query parameter.

Specify the type of job to return with the `job_type` query parameter.

URI to Get the Status of All Jobs

```
GET [api_version]/jobs
```

Curl Command to Get All Job Status

```
curl -i -X GET 'https://pce.my-company.com:8443/api/v2/orgs/1/jobs' -H 'Accept: application/json' -u $KEY:$TOKEN
```

Get a Job

URI to Get the Status of a Job

```
GET [api_version]/jobs/[href]
```

Curl Command to Get a Job Status

```
curl -i -X GET 'https://pce.my-company.com:8443/api/v2/orgs/1/jobs/[href]' -H 'Accept: application/json' -u $KEY:$TOKEN
```


Response Properties

Poll the HREF provided in the Location field of the response using the duration specified in Retry-After until the status is either "done" or "failed":

Property	Description	Type	In Results
href	HREF for resource	String	Yes
job_type	Query type defined during job creation	String	Yes
description	Reference information	String	Might not be in results
result	Query result	Object (HREF, not required)	Yes
requested_at	Time PCE received request	Date-time	Yes
requested_by	User who initiated request	Object (HREF, required)	Yes
terminated_at	Termination time of job (regardless of outcome)	Date-time	Yes
status	Status of the asynchronous request	Enum ("done", "pending", "running", or "failed")	Yes
created_by	Creator of request	Object (HREF, required)	Yes

Response - Updated Job Status

If the job is still running, the response includes a status of "running", as highlighted in blue below:

```
{
  "href": "/orgs/1/jobs/43f6e9e3-6a68-4481-87c6-18fd096dafbe",
  "job_type": ":illumio/async_requests",
  "description": "/orgs/1/labels",
  "result": {
  },
  "status": "running",
  "requested_at": "2016-01-14 23:16:52.303166",
  "requested_by": {
    "href": "/users/1"
```

```
}  
}
```

Delete a Job

URI to Delete a Job

```
DELETE [api_version]/jobs/[href]
```

Curl Command to Delete a Job

```
curl -i -X DELETE 'https://pce.my-company.com:8443/api/v2/orgs/1/jobs/[href]' -u  
$KEY:$TOKEN
```

Get the Job Results

This example demonstrates how to get job results after polling job returns a status of "done".

The `uuid` path parameter is required. The `filename` path parameter is optional, it specifies the filename to save the job as.

URI to Get Job Results

```
GET [api_version][org_href]/datafiles/[uuid]
```

Curl Command to Get Job Results

```
curl -i -X GET 'https://yourcompany.com:1234/api/v2/orgs/1/datafiles/[uuid]' -H  
'Accept: application/json' -u $KEY:$TOKEN
```

Response with Results of Request

```
{  
  "href": "/orgs/1/jobs/43f6e9e3-6a68-4481-87c6-18fd096dafbe",  
  "job_type": ":illumio/async_requests",  
  "description": "/orgs/1/labels",  
  "result": {  
    "href": "/orgs/1/datafiles/[uuid]"
```

```
  },  
  "status": "done",  
  "requested_at": "2016-01-14 23:16:52.303166",  
  "terminated_at": "2016-01-14 23:17:05.223047",  
  "requested_by": {  
    "href": "/users/1"  
  }  
}
```

PCE Management

This chapter contains the following topics:

Product Version	76
Authentication Settings	77
Password Policy	78
Supercluster Leader	82
PCE Health	82
Node Availability	96
No Op	97
Events	97
Organization Settings	102
Container Clusters	107
Access Restrictions and Trusted Proxy IPs	125

As an Illumio administrator, use the APIs listed in this chapter to manage the Policy Compute Engine (PCE).

You can manage many aspects of the PCE through APIs, from authentication and passwords to PCE health.

Product Version

This API returns the current version of the PCE software.

URI to Get Product Version

```
GET [api_version]/product_version
```

Curl Command to Get Product Version

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/product_version -H "Accept: application/json" -u $KEY:$TOKEN
```

Example Response

The response body has a format similar to this example:

```
{
  "version": "19.3.0",
  "build": 12864,
  "long_display": "19.3.0-12864",
  "short_display": "19.3.0"
}
```

Authentication Settings

This Public Experimental API gets or updates the authentication settings for the login domain (organization).

Authentication Settings API Methods

Functionality	HTTP	URI
Get authentication settings	GET	[api_version]/authentication_settings
Update authentication settings	PUT	[api_version]/authentication_settings

Get Authentication Settings

Curl Command to Get Authentication Settings



NOTE:

The `org/:org_id/` path parameter is not specified in this command.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/authentication_settings -H "Accept: application/json" -u $KEY:$TOKEN
```

Example Default Response

```
200 OK

{ "authentication_type": "Local" }
```

Update Authentication Settings

Curl Command to Update Authentication Settings



NOTE:

The `org/:org_id/` path parameter is not specified in this command.

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/authentication_
settings/password_policy -H "Content-Type: application/json" -d '{"authentication_
settings": "SAML"}' u $KEY:$TOKEN
```

Request Properties

Parameter	Description	Type	Required
Local	Local authentication.	String	No
SAML	Authentication with SAML.	String	No

Example Request Body

```
{"authentication_settings": "SAML"}
```

Password Policy

This Public Experimental API gets or updates the domain password policy. A default password policy is created automatically when a new login domain (organization) is created. There is only one password policy per login domain, so the same password policy applies to all users.

Password Policy API Methods

Functionality	HTTP	URI
Get the password policy	GET	[api_version]/authentication_settings/password_policy
Update the password policy	PUT	[api_version]/authentication_settings/password_policy

Curl Command Get the Password Policy



NOTE:

The `org/:org_id/` path parameter is not specified in this command.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/authentication_
services/password_policy -H "Accept: application/json" -u $KEY:$TOKEN
```

Example Default Response: 200 OK

```
{
  "require_type_number": true,
  "require_type_lowercase": true,
  "require_type_uppercase": true,
  "require_type_symbol": false,
  "min_characters_per_type": 1,
  "min_length": 8,
  "min_changed_characters": 1,
  "history_count": 1,
  "expire_time_days": 0,
  "updated_at": "2019-09-20T03:40:00Z",
  "updated_by": null
}
```

Response Properties

Parameter	Description	Type	Required
<code>require_type_number</code>	If true, the password must contain a numerical digit.	Boolean	Yes
<code>require_type_lowercase</code>	If true, the password must contain a lowercase letter.	Boolean	Yes
<code>require_type_uppercase</code>	If true, the password must contain an uppercase letter.	Boolean	Yes
<code>require_type_symbol</code>	If true, the password must contain a symbol, for example: ! @ # \$ % ^ * ? \u0026 \u003c \u003e	Boolean	Yes
<code>min_characters_per_type</code>	Minimum number of characters for each character type.	Integer	Yes
<code>min_length</code>	Minimum password length.	Integer	Yes

Parameter	Description	Type	Required
min_changed_characters	Minimum number of changed characters for a new password. Minimum: 1 Maximum: 4	Integer	Yes
history_count	Number of old passwords to remember. Minimum: 1 Maximum: 24	Integer	Yes
expire_time_days	Number of days until the password expires. A value of 0 (zero) means the password never expires. Minimum: 0 Maximum: 99	Integer	Yes
updated_at	RFC-3339 date-time timestamp of when the password policy was last updated. Automatically recorded by the system.	date-time String	Yes
updated_by	The username of the person that last updated this password policy (null for the default password policy). Automatically recorded by the system.	String	Yes

Update the Password Policy

Curl Command Update the Password Policy



NOTE:

The `org/:org_id/` path parameter is not specified in this command.

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/authentication_
services/password_policy -H "Content-Type: application/json" -u $KEY:$TOKEN -d '
{"require_type_symbol": true, "expire_time_days": 90}
```

Request Properties

At least three of the four available character types must be true, otherwise a 406 Not Acceptable error message is returned.*

Parameter	Description	Type	Required
require_type_number	If true, the password must contain a numerical digit.	Boolean	*
require_type_lowercase	If true, the password must contain a lowercase letter.	Boolean	*
require_type_uppercase	If true, the password must contain an uppercase letter.	Boolean	*
require_type_symbol	If true, the password must contain a symbol, for example: !@#\$%^*? \u0026 \u003c \u003e	Boolean	*
min_characters_per_type	Minimum number of characters for each character type.	Integer	No
min_length	Minimum password length.	Integer	No
min_changed_characters	Minimum number of changed characters for new passwords. Minimum: 1 Maximum: 4	Integer	No
history_count	Number of old passwords to remember. Minimum: 1 Maximum: 24	Integer	No
expire_time_days	Number of days password expires. A value of 0 (zero) means the password never expires. Minimum: 0 Maximum: 99	Integer	No

Example Request Body

Only the parameters to change must be included in the request body.

```
{
  "require_type_number": true,
  "require_type_lowercase": true,
  "require_type_uppercase": true,
  "require_type_symbol": true,
  "min_characters_per_type": 1,
  "min_length": 8,
  "min_changed_characters": 1,
  "history_count": 1,
}
```

```
"expire_time_days": 90
}
```

Supercluster Leader

The Supercluster Leader Public Stable API method checks each PCE in a Supercluster and indicates which PCE is the leader.

About the Supercluster Leader API

This call is typically made by a customer's Global Server Load Balancer (GSLB) to monitor the health of the leader.

Possible results:

- If the API returns an HTTP 202 response, the cluster where you made this call is the leader.
- If the API returns an HTTP 404 response, then the cluster where you made this call is a member.

For more information, see the *PCE Supercluster Deployment Guide*.

Get Supercluster Leader

```
GET [api_version]/supercluster/leader
```

Curl Command Get Supercluster Leader

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/supercluster/leader  
-H "Accept: application/json" -u $KEY:$TOKEN
```

PCE Health

The Public Experimental Health Check API displays health information about a 4X2 Supercluster or a PCE virtual appliance.



NOTE:

This API is only available for Illumio Core PCE installed on-premises and is not available for Illumio Cloud customers.

About the PCE Health API

With this API, you can see the following health information:

- How long the PCE has been running, its runlevel, and overall health (normal, warning, or error).
- Each node hostname, IP address, uptime, runlevel, and whether the PCE software is running properly.
- Each node type (core or data), and which data node is the database replica and which is the primary database. The replication delay for the database replica is also displayed.
- Information about PCE service alerts, such as the number of degraded or failed services in the cluster, so you can see where service failures have occurred.

PCE Health API Method

Functionality	HTTP	URI
Check the health of the PCE.	GET	[api_version]/health

Check PCE Health

URI to Check PCE Health

```
GET [api_version]/health
```

Curl Command Check PCE Health

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/health -H 'Accept: application/json' -u $KEY:'TOKEN'
```

PCE Health Response Properties

Property	Description	Type
status	<p>Current health status of the PCE. Possible values:</p> <ul style="list-style-type: none"> • normal: When a PCE health is a normal state it means: <ul style="list-style-type: none"> ◦ All required services are running. ◦ All nodes are running. ◦ CPU usage of all nodes is less than 95%. ◦ Memory usage of all nodes is less than 95%. ◦ Disk usage of all nodes is less than 95%. ◦ Database replication lag is less than or equal to 30 	String

Property	Description	Type
	<p>seconds.</p> <ul style="list-style-type: none"> • warning: When PCE health is in a warning state, it means: <ul style="list-style-type: none"> ◦ One or more nodes are unreachable. ◦ One or more optional services are missing, or one or more required services have been degraded. ◦ The CPU usage of any node is greater than or equal to 95%. ◦ Memory usage of any node is greater than or equal to 95%. ◦ Disk usage of any node is greater than or equal to 95%. ◦ Database replication lag is greater than 30 seconds. • critical: A PCE is considered to be in a critical state when one or more required services are missing. If a PCE enters a critical state, it might not be possible to authenticate to the PCE or get an API response depending on which services are missing from the PCE. 	
type	<p>The type of PCE:</p> <ul style="list-style-type: none"> • standalone: Indicates that this PCE is an on-premises 2x2 or 4x2 PCE cluster. Or one of the following types: • leader: Indicates that this PCE is the leader of a Super-cluster. • member: Indicates that this PCE is a member of a Super-cluster. 	String
fqdn	The fully qualified domain name (FQDN) of the PCE.	String
available_seconds	The length of time that this PCE has been available, measured in seconds.	Number
notifications	Health warnings related to the PCE, which contain the following properties:	

Property	Description	Type
	<ul style="list-style-type: none"> • status: Severity status of this notification. Possible values include: <code>normal</code>, <code>warning</code>, or <code>critical</code>. • token: Description of the notification. • message: Notification message. 	
<code>listen_only_mode_enabled_at</code>	<p>Indicates when listen-only mode was enabled for this PCE.</p> <p>For information about enabling or disabling listen-only mode for a PCE, see the <i>PCE Administration Guide</i>.</p>	String
<code>nodes</code>	<p>The nodes that comprise your PCE cluster. For each node of your PCE, this API call returns the following properties:</p> <ul style="list-style-type: none"> • hostname: The node hostname. • ip_address: The node IP address. • runlevel: (Number) The current runlevel of the PCE software on the node. For more information about runlevels and their usage, see the <i>PCE Administration Guide</i>. • uptime_seconds: Seconds since this node has been restarted. • cpu: Percentage of the node CPU being used. Includes the following two sub-properties: <ul style="list-style-type: none"> ◦ status: Either <code>normal</code>, <code>warning</code>, or <code>critical</code>. ◦ percent: (Number) Percentage of the node CPU being used. • disk: Percentage of the node's disk that is being used. Includes the following two sub-properties: <ul style="list-style-type: none"> ◦ status: Either <code>normal</code>, <code>warning</code>, or <code>critical</code>. ◦ percent: (Number) Percentage of the node disk being used. • memory: Percentage of the node's memory that is being used. Includes the following two sub-properties: <ul style="list-style-type: none"> ◦ status: Either <code>normal</code>, <code>warning</code>, or <code>critical</code>. ◦ percent: (Number) Percentage of the node disk being used. • services: The status of all PCE services running on the 	String

Property	Description	Type
	<p>node. Possible status for PCE services include:</p> <ul style="list-style-type: none"> ◦ <code>running</code>: The service is fully running and operational. ◦ <code>not running</code>: The service has stopped running. ◦ <code>partial</code>: The service is running but in a partial state. ◦ <code>optional</code> ◦ <code>unknown</code> <ul style="list-style-type: none"> • <code>generated_at</code>: Timestamp when this information was generated. 	
network	<p>PCE 2x2 or 4x2 Deployment</p> <p>For a PCE 2x2 or 4x2 deployment, the <code>network</code> property provides latency information between the database primary and database replica data nodes in your PCE for policy and traffic data.</p> <p>This property also indicates which data node in your PCE is the database primary database and which is the database replica.</p> <p>This type of database replication is called <code>intracluster</code> in the REST API.</p> <p>Sub-properties include:</p> <p><code>replication</code>: The category of properties that provide database replication latency information for a PCE cluster. (For a PCE Supercluster, this information is provided for each PCE in the Supercluster.)</p> <ul style="list-style-type: none"> • <code>type</code>: Type of replication. <code>intracluster</code> for a PCE 2x2 or 4x2 deployment. • <code>details</code>: Includes the following properties: <ul style="list-style-type: none"> ◦ <code>database_name</code>: Either <code>agent</code> for policy data or <code>traffic</code> for traffic data. ◦ <code>primary_fqdn</code>: The FQDN of the database primary node. ◦ <code>replica_fqdn</code>: FQDN of the replica database node. • <code>value</code>: The amount of replication lag between the primary and database replica for both policy and traffic data. 	Array

Property	Description	Type
	<ul style="list-style-type: none"> ◦ <code>status</code>: Either <code>normal</code>, <code>warning</code>, or <code>critical</code>. ◦ <code>lag_seconds</code>: The amount of lag measured in seconds between the primary and replica databases for both policy and traffic data. <p>Supercluster Deployment</p> <p>If you have deployed a PCE Supercluster, the PCE health call also returns information about the database replication between the PCE you are currently logged into and all other PCEs in the Supercluster.</p> <p>In a Supercluster deployment, the security policy provisioned on the leader is replicated to all other PCEs in the Supercluster. Additionally, all PCEs in the Supercluster (leader and members) replicate copies of each workload's context, such as IP addresses, to all other PCEs in the Supercluster.</p> <p>This other type of database replication for a Supercluster is called <code>intercluster</code> in the REST API, and information is provided for all PCEs in the Supercluster.</p> <p>Properties include:</p> <p><code>replication</code>: The category of properties that provide database replication latency information for a PCE cluster.</p> <ul style="list-style-type: none"> • <code>type</code>: Type of replication. <code>intercluster</code> for a PCE Supercluster deployment. • <code>details</code>: Includes the following properties: <ul style="list-style-type: none"> ◦ <code>fqdn</code>: The FQDN of the database primary of the other PCEs listed in this section. • <code>value</code>: The amount of replication lag between the PCE you are logged into and one of the other PCEs in the Supercluster. <ul style="list-style-type: none"> ◦ <code>status</code>: Either <code>normal</code>, <code>warning</code>, or <code>critical</code>. ◦ <code>lag_seconds</code>: The amount of lag measured in seconds between the PCE you are logged into and the other PCE listed in this section. 	
<code>generated_at</code>	The timestamp of when the information was generated.	String

PCE Health Response

Example response returned from the PCE Health API.

```
[
  {
    "status": "normal",
    "type": "standalone",
    "fqdn": "pce.mycompany.com",
    "available_seconds": 84133,
    "notifications": [],
    "listen_only_mode_enabled_at": null,
    "nodes": [
      {
        "hostname": "pce_core1.mycompany.com",
        "ip_address": "192.0.1.0",
        "type": "core",
        "runlevel": 5,
        "uptime_seconds": 2051301,
        "cpu": {
          "status": "normal",
          "percent": 7
        },
        "disk": [
          {
            "location": "disk",
            "value": {
              "status": "normal",
              "percent": 17
            }
          }
        ],
        "memory": {
          "status": "warning",
          "percent": 85
        },
        "services": {
          "status": "normal",
          "services": {
            "running": [
```



```
        "agent_background_worker_service",
        "agent_service",
        "agent_traffic_service",
        "auditable_events_service",
        "collector_service",
        "ev_service",
        "executor_service",
        "fluentd_source_service",
        "login_service",
        "memcached",
        "node_monitor",
        "search_index_service",
        "server_load_balancer",
        "service_discovery_server",
        "traffic_worker_service",
        "web_server",
        "nfc_service"
    ]
}
},
"generated_at": "2020-03-03T19:38:52+00:00"
},
{
    "hostname": "pce_core2.mycompany.com",
    "ip_address": "192.0.2.0",
    "type": "core",
    "runlevel": 5,
    "uptime_seconds": 2051226,
    "cpu": {
        "status": "normal",
        "percent": 7
    },
    "disk": [
        {
            "location": "disk",
            "value": {
                "status": "normal",
                "percent": 16
            }
        }
    ]
}
```

```
        }
      }
    ],
    "memory": {
      "status": "warning",
      "percent": 81
    },
    "services": {
      "status": "normal",
      "services": {
        "running": [
          "agent_background_worker_service",
          "agent_service",
          "agent_traffic_service",
          "auditable_events_service",
          "collector_service",
          "ev_service",
          "executor_service",
          "fluentd_source_service",
          "login_service",
          "memcached",
          "node_monitor",
          "search_index_service",
          "service_discovery_server",
          "traffic_worker_service",
          "web_server"
        ]
      }
    },
    "generated_at": "2020-03-03T19:38:30+00:00"
  },
  {
    "hostname": "pce_core3.mycompany.com",
    "ip_address": "192.0.3.0",
    "type": "core",
    "runlevel": 5,
    "uptime_seconds": 2051192,
    "cpu": {
```

```
    "status": "normal",
    "percent": 7
  },
  "disk": [
    {
      "location": "disk",
      "value": {
        "status": "normal",
        "percent": 16
      }
    }
  ],
  "memory": {
    "status": "warning",
    "percent": 90
  },
  "services": {
    "status": "normal",
    "services": {
      "running": [
        "agent_background_worker_service",
        "agent_service",
        "agent_traffic_service",
        "auditable_events_service",
        "collector_service",
        "ev_service",
        "executor_service",
        "fluentd_source_service",
        "login_service",
        "memcached",
        "node_monitor",
        "search_index_service",
        "service_discovery_server",
        "traffic_worker_service",
        "web_server"
      ]
    }
  },
}
```

```
    "generated_at": "2020-03-03T19:38:48+00:00"
  },
  {
    "hostname": "pce_core4.mycompany.com",
    "ip_address": "192.0.4.0",
    "type": "core",
    "runlevel": 5,
    "uptime_seconds": 2051136,
    "cpu": {
      "status": "normal",
      "percent": 6
    },
    "disk": [
      {
        "location": "disk",
        "value": {
          "status": "normal",
          "percent": 16
        }
      }
    ],
    "memory": {
      "status": "warning",
      "percent": 84
    },
    "services": {
      "status": "normal",
      "services": {
        "running": [
          "agent_background_worker_service",
          "agent_service",
          "agent_traffic_service",
          "auditable_events_service",
          "collector_service",
          "ev_service",
          "executor_service",
          "fluentd_source_service",
          "login_service",
```

```
        "memcached",
        "node_monitor",
        "search_index_service",
        "server_load_balancer",
        "service_discovery_server",
        "traffic_worker_service",
        "web_server"
    ]
}
},
"generated_at": "2020-03-03T19:38:51+00:00"
},
{
    "hostname": "pce_datae0.mycompany.com",
    "ip_address": "192.0.5.0",
    "type": "data0",
    "runlevel": 5,
    "uptime_seconds": 2051052,
    "cpu": {
        "status": "normal",
        "percent": 41
    },
    "disk": [
        {
            "location": "disk",
            "value": {
                "status": "normal",
                "percent": 19
            }
        }
    ],
    "memory": {
        "status": "normal",
        "percent": 26
    },
    "services": {
        "status": "normal",
        "services": {
```

```
        "running": [
            "agent_traffic_redis_cache",
            "agent_traffic_redis_server",
            "citus_database_service",
            "database_monitor",
            "database_service",
            "fileservice_service",
            "flow_analytics_service",
            "fluentd_data_service",
            "node_monitor",
            "service_discovery_server",
            "set_server_redis_server",
            "traffic_query_service"
        ]
    },
    "generated_at": "2020-03-03T19:38:21+00:00"
},
{
    "hostname": "pce_datae1.mycompany.com",
    "ip_address": "192.0.6.0",
    "type": "data1",
    "runlevel": 5,
    "uptime_seconds": 2050979,
    "cpu": {
        "status": "normal",
        "percent": 2
    },
    "disk": [
        {
            "location": "disk",
            "value": {
                "status": "normal",
                "percent": 21
            }
        }
    ],
    "memory": {
```

```
        "status": "normal",
        "percent": 21
    },
    "services": {
        "status": "normal",
        "services": {
            "running": [
                "agent_traffic_redis_cache",
                "citus_database_replica_service",
                "database_monitor",
                "database_replica_service",
                "fileserver_replica_service",
                "flow_analytics_service",
                "fluentd_data_service",
                "node_monitor",
                "service_discovery_agent",
                "traffic_query_service"
            ]
        }
    },
    "generated_at": "2020-03-03T19:38:02+00:00"
}
],
"network": {
    "replication": [
        {
            "type": "intracluster",
            "details": {
                "database_name": "agent",
                "primary_fqdn": "bkhorram-qa-6node-v0-pce-1-dbase0"
            },
            "value": {
                "status": "normal",
                "lag_seconds": 0
            }
        },
        {
            "type": "intracluster",
```

```
    "details": {
      "database_name": "traffic",
      "primary_fqdn": "bkhorrarn-qa-6node-v0-pce-1-dbase0"
    },
    "value": {
      "status": "normal",
      "lag_seconds": 0
    }
  }
]
},
"generated_at": "2020-03-03T19:38:52+00:00"
}
```

Node Availability

This Public Stable API method allows the load balancer to monitor the health of the PCE core nodes in a 2x2 or 4x2 cluster. This feature is only available if the PCE is deployed as software in your datacenter.



NOTE:
This API call does not require authentication.

URI to Check Node Availability

```
GET [api_version]/node_available
```

Curl Command to Check Node Availability

-X GET and authentication are not required for this method. The curl -v flag provides verbose output.

```
curl -v https://pce.my-company.com:8443/api/v2/node_available
```

Or, you can use -i -X GET to return a 200 OK status if the node is available:

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/node_available
```


Example Response

Returns 200 OK if the core node is healthy, and it can see at least one of each service running in the PCE cluster.

Otherwise, it returns a 404 error.

For example, if the PCE is healthy and accessible, the response is 200 OK.

Run a Health Check from a Load Balancer

In a production deployment, customers run health checks from a Load Balancer. The actual request syntax varies, but here is a sample command for Infoblox:

```
GET /api/v2/node_available HTTP/1.1
```

No Op

The No Op Public Stable API makes a call to the PCE without performing any operations. This API is used to check connectivity to and from the PCE. Use this API to verify that new authentication credentials are working after creating a new set of keys.

URI for No Op

```
GET [api_version]/noop
```

Curl Command for No Op

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/noop -H "Accept: application/json" -u $KEY:'TOKEN'
```

Events

This Public Experimental API gets a collection of events or an individual event.



NOTE:

Starting with Illumio Core 18.2, use this Events API instead of Audit Events.

Events include logging a user in or out of the PCE, granting a role to a user, pairing or unpairing a workload, creating a label, ruleset, or IP list.

Event Types

For a complete list of JSON events, descriptions, CEF/LEEF success events, and CEF/LEEF failure events, see the *Events Administration Guide*.

Event API Methods

Functionality	HTTP	URI
Get a collection of events	GET	[api_version][org_href]/events
Get an individual event	GET	[api_version][event_href]

Get Events

This API gets a collection of events or a specific event identified by an event ID (in the form of a UUID).

Get Events Collection

When getting a collection of events, be aware of the following caveats:

- Use the `max_results` query parameter to increase the maximum number of events returned.
- The largest value accepted for `max_results` is 10000. To return more than 10000 events, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Events

```
GET [api_version][org_href]/events
```

URI to Get an Individual Event

```
GET [api_version][event_href]
```

Parameters

Parameter	Description	Type
<code>xorg_id</code>	Organization ID in which the event occurred.	Integer
<code>created_by</code>	Information about the person, agent, or system that created the event. Created by <i>system</i> : <ul style="list-style-type: none"> • <code>system</code>: Appears only if the event was generated by the PCE. 	String

Parameter	Description	Type
	Created by <i>user</i> properties: <ul style="list-style-type: none"> • href: URI of the user who created the event. • username: The user's name (usually formatted as an e-mail address). Created by <i>workload</i> properties: <ul style="list-style-type: none"> • href: URI of the agent on the workload that initiated the event. • hostname: The hostname of the workload. 	
event_type	Type of the event specified by the event_type query parameter if given. If no query parameters are given, all event types are returned. For types of events returned from a GET call, see the response properties in the table below.	String
status	Status of the event, either <i>success</i> or <i>failure</i> .	String
timestamp	Timestamp.	Hash
timestamp [gte]	Event start timestamp in RFC 3339 format.	String
timestamp [lte]	Event end timestamp in RFC 3339 format.	String
severity	Severity level of the events retrieved. Values include: <ul style="list-style-type: none"> • Warning (<i>warning</i>): A warning that the event is likely to occur if action is not taken. • Error (<i>err</i>) • Information (<i>info</i>): Normal operational messages, which can be harvested for reporting and measuring throughput; for example, [a user pairing or unpairing workloads in the PCE web console. 	String
max_results	Maximum number of events to return. The default is 100, and the maximum is 10000.	Integer

Curl Command to Get an Event

You need the ID of the system event you want to get, which is the number at the end of its HREF path property: `"/2/events/68632"`.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/events/12345 -H
"Accept: application/json" -u $KEY:$TOKEN
```

Curl Command Get Event Collection

In this example, only two events are returned because of `max_events=2`.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/events?max_results=2
-H "Accept: application/json" -u $KEY:$TOKEN
```

Example Response

```
[
  {
    "href": "/orgs/1/events/xxxxxx-5f59-46ab-8f18-xxxxxxx",
    "timestamp": "2019-09-03T01:xx:xx.xxxZ",
    "pce_fqdn": "pce.my-company.com",
    "created_by": {
      "agent": {
        "href": "/orgs/1/agents/xxx",
        "hostname": "xxx-xxxxx-xxxx"
      }
    },
    "event_type": "agent.clone_detected",
    "status": null,
    "severity": "info",
    "action": null,
    "resource_changes": [],
    "notifications": [
      {
        "uuid": "xxxxxx-e04b-43bc-a64a-xxxxxxxx",
        "notification_type": "agent.clone_detected",
        "info": {
          "agent": {
            "href": "/orgs/1/agents/xxx",

```

```
        "name": null,
        "hostname": "xxx-xxxxx-xxxx"
      }
    }
  ]
},
{
  "href": "/orgs/1/events/xxxxxxx-60a2-4db4-b0f4-xxxxxxxxxxx",
  "timestamp": "2019-09-03T0x:xx:xx.xxxZ",
  "pce_fqdn": "pce.my-company.com",
  "created_by": {
    "agent": {
      "href": "/orgs/1/agents/xxx",
      "hostname": "xxx-xxxxx-xxxx"
    }
  },
  "event_type": "agent.clone_detected",
  "status": null,
  "severity": "info",
  "action": null,
  "resource_changes": [],
  "notifications": [
    {
      "uuid": "xxxxxxxx-4833-4975-bf9d-xxxxxxxxxxxxx",
      "notification_type": "agent.clone_detected",
      "info": {
        "agent": {
          "href": "/orgs/1/agents/xxx",
          "name": null,
          "hostname": "xxx-xxxxx-xxxx"
        }
      }
    }
  ]
}
]
```

Organization Settings

For Organization Settings parameters, properties, JSON request and response bodies, and example curl commands, see "Organization Settings" in the [Illumio Core REST API Reference](#).

Get Events Settings

Returns events settings information.

For parameters, properties, JSON response body, and example curl command, see "Get Events Settings" in the [Illumio Core REST API Reference](#).

Example JSON Response Body for Get Events Settings

```
{
  "audit_event_retention_seconds": 180,
  "audit_event_min_severity": "information",
  "format": "JSON"
}
```

Update Events Settings

Updates event settings.

For parameters, properties, JSON request body, and example curl command, see "Update Events Settings" in the [Illumio Core REST API Reference](#).

Example JSON Request Body for Update Events

```
{
  "audit_event_retention_seconds": 90,
  "audit_event_min_severity": "information"
}
```

Syslog Destinations

Use this API to specify a local syslog location and/or one or more remote syslog locations.

Get Syslog Destinations

Returns all syslog destination information.

For parameters, properties, JSON response body, and example curl command, see "Get Syslog Destinations" in the [Illumio Core REST API Reference](#).

Example JSON Response Body with Local and Remote Syslog Location Information

```
[
  {
    "href": "/api/v2/orgs/1/settings/syslog/destinations/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "pce_scope": [ "my-company0.com", "my-company1.com", "my-company2.com" ],
    "type": "local_syslog",
    "description": "local dev/log",
    "audit_event_logger": {
      "configuration_event_included": true,
      "system_event_included": true,
      "min_severity": "information"
    },
    "traffic_event_logger": {
      "traffic_flow_allowed_event_included": true,
      "traffic_flow_potentially_blocked_event_included": true,
      "traffic_flow_blocked_event_included": true
    },
    "node_status_logger": {
      "node_status_included": true
    }
  },
  {
    "href": "/api/v2/orgs/1/settings/syslog/destinations/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "pce_scope": [ "remote-my-company0.com", "remote-my-company1.com" ],
    "type": "remote_syslog",
    "description": "remotesyslog",
    "audit_event_logger": {
      "configuration_event_included": true,
      "system_event_included": false,
      "min_severity": "warning"
    },
    "traffic_event_logger": {
      "traffic_flow_allowed_event_included": true,
      "traffic_flow_potentially_blocked_event_included": true,

```

```
        "traffic_flow_blocked_event_included": true
    },
    "node_status_logger": {
        "node_status_included": true
    },
    "remote_syslog": {
        "address" : "my-company-20.com",
        "port"    : 12345,
        "protocol" : 6,
        "tls_enabled" : false,
        "tls_verify_cert" : false
    }
}
]
```

Get a Syslog Destination

Returns information about one syslog destination.

For parameters, properties, JSON response body, and example curl command, see "Get a Syslog Destination" in the [illumio Core REST API Reference](#).

Example JSON Response Body with Remote Syslog Location Information

```
{
  "href": "/api/v2/orgs/1/settings/syslog/destinations/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "pce_scope": [ "remote-my-company0.com", "remote-my-company1.com" ],
  "type": "remote_syslog",
  "description": "remotesyslog",
  "audit_event_logger": {
    "configuration_event_included": true,
    "system_event_included": false,
    "min_severity": "warning"
  },
  "traffic_event_logger": {
    "traffic_flow_allowed_event_included": true,
    "traffic_flow_potentially_blocked_event_included": true,
    "traffic_flow_blocked_event_included": true
  },
}
```



```
"node_status_logger": {
  "node_status_included": true
},
"remote_syslog": {
  "address" : "my-company-20.com",
  "port"    : 12345,
  "protocol" : 6,
  "tls_enabled" : false,
  "tls_verify_cert" : false
}
}
```

Create a Syslog Destination

Creates a local and remote syslog destination.

For parameters, properties, JSON request body, and example curl command, see "Create a Syslog Destination" in the [Illumio Core REST API Reference](#).

Example JSON Request Body to Create a Remote Syslog Destination

```
{
  "pce_scope": [ "my-company0.com", "my-company1.com", "my-company2.com" ],
  "type": "remote_syslog",
  "description": "remote syslog",
  "audit_event_logger": {
    "configuration_event_included": true,
    "system_event_included": false,
    "min_severity": "warning"
  },
  "traffic_event_logger": {
    "traffic_flow_allowed_event_included": true,
    "traffic_flow_potentially_blocked_event_included": true,
    "traffic_flow_blocked_event_included": true
  },
  "node_status_logger": {
    "node_status_included": true
  },
  "remote_syslog": {
    "address" : "my-company-20.com",
```

```
    "port"      : 12345,  
    "protocol" : 6,  
    "tls_enabled" : false,  
    "tls_verify_cert" : false  
  }  
}
```

Update a Syslog Destination

Updates a local and a remote syslog destination.

For parameters, properties, JSON request body, and example curl command, see "Update a Syslog Destination" in the [Illumio Core REST API Reference](#).

Example JSON Request Body to Update a Syslog Destination

```
{  
  "href": "/api/v2/orgs/1/settings/syslog/destinations/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
  "pce_scope": [ "my-company0.com", "my-company1.com", "my-company2.com" ],  
  "type": "remote_syslog",  
  "description": "localhost syslog",  
  "audit_event_logger": {  
    "configuration_event_included": true,  
    "system_event_included": true,  
    "min_severity": "information"  
  },  
  "traffic_event_logger": {  
    "traffic_flow_allowed_event_included": true,  
    "traffic_flow_potentially_blocked_event_included": true,  
    "traffic_flow_blocked_event_included": true  
  },  
  "node_status_logger": {  
    "node_status_included": false  
  },  
  "remote_syslog": {  
    "address" : "my-company-20.com",  
    "port"      : 67890,  
    "protocol" : 6,  
    "tls_enabled" : false,  
  }  
}
```

```

        "tls_verify_cert" : false
    }
}
    
```

Delete a Syslog Destination

Deletes a syslog destination.

For parameters, properties, and example curl command, see "Delete a Syslog Destination" in the [Illumio Core REST API Reference](#).

Container Clusters

The Illumio Core uses three groups of APIs to manage container clusters:

- Container Cluster API (GET, POST, PUT, DELETE)
- Container Cluster Workload Profiles API (GET, POST, PUT, DELETE)
- Container Cluster Service Backend API (GET)

For more information, see the Illumio Core for Kubernetes and OpenShift guide.

Container Cluster API

A container cluster object is used to store all the information about a Kubernetes cluster in the PCE by collecting telemetry from Kubelink. Each Kubernetes cluster maps to one container cluster object in the PCE.

Use these methods to get, create, update, or delete container clusters:

Functionality	HTTP	URI
Get the list of container clusters	GET	GET /orgs/:xorg_id/container_clusters
Get the specified container cluster	GET	GET /orgs/:xorg_id/container_clusters/:uuid
Create a container cluster	POST	POST /orgs/:xorg_id/container_clusters
Update the specified container cluster	PUT	PUT /orgs/:xorg_id/container_clusters/:uuid
Delete the specified container cluster	DELETE	DELETE /orgs/:xorg_id/container_clusters/:uuid

Query Parameters for the GET Method

Use the following required and optional parameters:

Parameter	Description	Type	Required
href	URI of the container cluster.	String	Yes
name	User assigned name of the container cluster.	String	Yes
description	User-assigned description of the container cluster.	String	Yes
nodes		Array	No
machine_id	This parameter has the following property: <ul style="list-style-type: none"> pod_subnet: The pod subnet 	Object String	Yes
manager_type	Manager of the container cluster (and version).	String	No
network_type	Type of network.	String	No
last_connected	Date-time format.	String	No
online	Online: true/false.	Boolean	No
errors	The object error_type has the following properties: <ul style="list-style-type: none"> audit_event: <ul style="list-style-type: none"> href duplicate_ids error_type 	Array Object String Array String String	No
kubelink_version	Kubelink software version.	String	No
pce_fqdn	PCE FQDN for this container cluster; used only in Supercluster.	String	No

Query Parameters for the POST and PUT Methods

Use the following parameters:

Parameter	Description	Type	Required
name	User-assigned name of the cluster	String	Yes
description	User-assigned description of the cluster	String	No

Curl Examples and Responses

Curl Command for GET

```
curl --request GET --url https://pce.my-company.com:8443/api/v2/orgs/1/container_
clusters --header 'authorization: Basic
YXBpXzE2YjBkYjI0MjJhZGNkYWU5OjA5ZmRjNjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMD
AzZmMzOTQ1ZGMxYzEwZGZlMzJm='
```

Example Response for GET

```
[
  {
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f",
    "pce_fqdn":null,
    "name":"k8s2",
    "description":"",
    "manager_type":"Kubernetes v1.16.2",
    "last_connected":"2019-10-28T22:48:31.228Z",
    "kubelink_version":"2.0.0-master.96e58b",
    "online":true,
    "nodes":
    [
      [
        {
          "name":"node1",
          "pod_subnet":"10.233.64.0/24"
        },
        {
          "name":"node2",
          "pod_subnet":"10.233.65.0/24"
        },
        {
          "name":"node3",
          "pod_subnet":"10.233.66.0/24"
        }
      ],
      "errors":[]
    },
    {
      "href":"/orgs/1/container_clusters/ad678193-8e2f-402b-a864-4947dcc0c6d7",
```

```
"pce_fqdn":null,
"name":"Openshift 3.11",
"description:"",
"manager_type":"Openshift v3.11.43",
"last_connected":"2019-10-28T22:50:30.201Z",
"kubelink_version":"1.0.0-master.a81280",
"online":true,
"nodes":
[
  {
    "name":"ip-172-31-19-198.us-west-2.compute.internal",
    "pod_subnet":"10.128.0.0/23"
  },
  {
    "name":"ip-172-31-20-168.us-west-2.compute.internal",
    "pod_subnet":"10.131.0.0/23"
  },
  {
    "name":"ip-172-31-22-56.us-west-2.compute.internal",
    "pod_subnet":"10.130.0.0/23"
  },
  {
    "name":"ip-172-31-27-241.us-west-2.compute.internal",
    "pod_subnet":"10.129.0.0/23"
  }
],
"errors":[]
},
{
  "href": "/orgs/1/container_clusters/bef57e90-97d4-4744-a129-5d35aa12b21b",
  "pce_fqdn":null,
  "name":"k8s3 Cluster",
  "description":"Flannel Vx Lan",
  "manager_type":"Kubernetes v1.13.2",
  "last_connected":"2019-10-28T22:47:59.122Z",
  "kubelink_version":"EYE-60264",
  "online":true,
  "nodes":
```

```
[
  {
    "name": "k8s3master",
    "pod_subnet": "10.244.0.0/24"
  },
  {
    "name": "k8s3minion1",
    "pod_subnet": "10.244.2.0/24"
  },
  {
    "name": "k8s3minion2",
    "pod_subnet": "10.244.1.0/24"
  }
],
"errors": []
},
{
  "href": "/orgs/1/container_clusters/d7d62400-7650-4407-ae9b-71803dbb1324",
  "pce_fqdn": null,
  "name": "k8s1 v4",
  "description": "",
  "manager_type": "Kubernetes v1.12.4",
  "last_connected": "2019-10-24T23:58:55.795Z",
  "kubelink_version": "EYE-61567",
  "online": false,
  "nodes":
  [
    {
      "name": "k8s1master",
      "pod_subnet": "10.244.0.0/24"
    },
    {
      "name": "k8s1minion1",
      "pod_subnet": "10.244.2.0/24"
    },
    {
      "name": "k8s1minion2",
      "pod_subnet": "10.244.1.0/24"
    }
  ]
}
```

```
    }
  ],
  "errors": []
}
]
```

Curl Example for POST

```
curl --request POST --url https://pce.my-company.com:8443/api/v2/orgs/1/container_
clusters --header 'authorization: Basic
jI0MjJhZGNkYWU50jA5ZmRjNjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMDAzZmMzOTQ1ZGM
xYzEwZGJhZTg5NzlmZjM=' --header 'content-type: application/json' --data '{"name":
"test","description": "test"}'
```

Curl Example for PUT

```
curl --request PUT --url https://pce.my-company.com:8443/api/v2/orgs/1/container_
clusters/1b851d4b-f22d-47be-b744-f3c2dca490a0 --header 'authorization: Basic
YXBpXzE2YjBkYjI0MjJhZGNkYWU50jA5ZmRjNjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMD
AzZmMzOTQ1ZGMxYzEwZGJhZTg5NzlmZjM=' --header 'content-type: application/json' --
data '{"name": "test","description": "test"}'
```

Example Response for POST

```
{
  "href": "/orgs/1/container_clusters/1b851d4b-f22d-47be-b744-f3c2dca490a0",
  "pce_fqdn": null,
  "name": "test",
  "description": "test",
  "manager_type": null,
  "last_connected": null,
  "kubelink_version": null,
  "online": false,
  "nodes": [],
  "errors": [],
  "container_cluster_token": "1_
0dfec0acb8e4bc53e052874874da0c24e7ac98da3b3954e3c9ea6f9860722e84"
}
```


Container Cluster Workload Profiles API

When you install an Illumio VEN on a container cluster, all pods in the container cluster are unmanaged or not visible in the PCE. However, all namespaces that exist on the container clusters are reported by Kubelink and made visible via the Container Container Workload Profiles API.

Each container workload profile maps to a Kubernetes namespace and can be either managed or unmanaged. The default state for a profile is unmanaged.

Use these methods to get, create, update, or delete container cluster workload profiles:

Functionality	HTTP	URI
Get the list of container cluster workload profiles	GET	GET /orgs/:xorg_id/container_clusters/: container_cluster_id/container_workload_profiles
Create container cluster workload profiles	POST	POST /orgs/:xorg_id/container_clusters/: container_cluster_id/container_workload_profiles
Update the specified container cluster workload profile	PUT	PUT /orgs/:xorg_id/container_clusters/: container_cluster_id/container_workload_profiles/:container_workload_profile_id
Delete the specified container cluster workload profile	DELETE	DELETE /orgs/:xorg_id/container_clusters/: container_cluster_id/container_workload_profiles/:container_workload_profile_id

Query Parameters for GET Method



NOTE: The `assign_labels` property was deprecated and the new `labels` property was added.

Parameter	Description	Type	Required
<code>href</code>	URI of the container cluster workload.	String	Yes
<code>name</code>	A friendly name given to a profile if the namespace is not user-friendly.	String	Yes
<code>namespace</code>	Namespace string to match. Supports partial matches.	String	Yes
<code>Description</code>	Description of the profile.	String	No

Parameter	Description	Type	Required
assign_labels (deprecated)	Labels to assign to the workload that matches the namespace.		No
labels	Labels to assign to the workload that matches the namespace.	Array	
mode	Filter by mode. Has three modalities: unmanaged, illuminated, enforced	String	No
log_traffic (deprecated)	Return container workload profiles with log traffic true or false. Specify true if you want to log traffic events from this workload.	Boolean	No
linked	Manager for this container cluster (and version).	String	No
created_at	Timestamp when this profile was created.	String	No
created_by	Date-time format. <ul style="list-style-type: none"> href: The user who created this profile. 	Object String	Yes
updated_by	Date-time format. <ul style="list-style-type: none"> href: The user who last updated this profile. 	Object String	Yes
updated_at	Date-time format.	String	No

Query Parameters for PUT and POST Methods



NOTE: The `assign_labels` property was deprecated and the new `labels` property was added.

Parameters	Description	Type	Required
name	A friendly name given to a profile if the namespace is not user-friendly.	String	Yes
description	Description of the profile.	String	No

Parameters	Description	Type	Required
assign_labels (deprecated)	Labels to assign to the workload that matches the namespace.		No
labels	Labels to assign to the workload that matches the namespace.	Array	
mode	Filter by mode. Has three modalities: unmanaged, illuminated, enforced	String	No
log_traffic (deprecated)	Return container workload profiles with log traffic true or false. Specify true if you want to log traffic events from this workload.	Boolean	No

Curl Examples and Responses

Curl example for GET

```
curl --request GET --url https://pce.my-company.com:8443/api/v2/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles --header 'authorization: Basic NjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMDAzZmMzOTQ1ZGMxYzEwZGJhZTg5NzlmZjM=' --header 'content-type: application/json'
```

Example Response for GET

```
[
  {
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles/f68d2408-0f46-4ebb-89ff-9c1c66ef4944",
    "name":"Default Profile",
    "namespace":null,
    "mode":"illuminated",
    "log_traffic":false,
    "visibility_level":"flow_summary",
    "assign_labels":[
      {
```

```
    "href":"/orgs/1/labels/18"
  },
  {
    "href":"/orgs/1/labels/22"
  }
],
"linked":false,
"created_at":"2019-10-25T19:56:00.763Z",
"created_by":{
  "href":"/users/1"
},
"updated_at":"2019-10-25T19:58:10.279Z",
"updated_by":{
  "href":"/users/1"
}
},
{
  "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles/62ba67e6-6696-41ab-b5bd-558eea933eec",
  "namespace":"kube-node-lease",
  "mode":"illuminated",
  "log_traffic":false,
  "visibility_level":"flow_summary",
  "assign_labels":[
    {
      "href":"/orgs/1/labels/18"
    },
    {
      "href":"/orgs/1/labels/22"
    }
  ],
  "linked":true,
  "created_at":"2019-10-25T19:58:39.483Z",
  "created_by":{
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f"
  },
  "updated_at":"2019-10-25T19:58:39.483Z",
  "updated_by":{
```

```
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f"
  }
},
{
  "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles/7b068f40-918b-4ea4-b7e3-2b988e37f6b2",
  "namespace":"kube-public",
  "mode":"illuminated",
  "log_traffic":false,
  "visibility_level":"flow_summary",
  "assign_labels":[
    {
      "href":"/orgs/1/labels/18"
    },
    {
      "href":"/orgs/1/labels/22"
    }
  ],
  "linked":true,
  "created_at":"2019-10-25T19:58:39.502Z",
  "created_by":{"
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f"
  },
  "updated_at":"2019-10-25T19:58:39.502Z",
  "updated_by":{"
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f"
  }
},
{
  "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles/d19e0634-5c3e-457d-9e14-2e0570d86cd9",
  "namespace":"kube-system",
  "description":"",
  "mode":"illuminated",
  "log_traffic":false,
  "visibility_level":"flow_summary",
  "assign_labels":[
    {
```

```

        "href":"/orgs/1/labels/18"
    },
    {
        "href":"/orgs/1/labels/22"
    },
    {
        "href":"/orgs/1/labels/33"
    }
],
"linked":true,
"created_at":"2019-10-25T19:58:39.522Z",
"created_by":{
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f"
},
"updated_at":"2019-10-25T21:01:57.527Z",
"updated_by":{
    "href":"/users/1"
}
},
{
    "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles/7524d406-b3f3-4532-ae50-06d5dc2da13b",
    "namespace":"default",
    "description":"",
    "mode":"illuminated",
    "log_traffic":false,
    "visibility_level":"flow_summary",
    "assign_labels":[

    ],
    "linked":true,
    "created_at":"2019-10-25T19:58:39.434Z",
    "created_by":{
        "href":"/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f"
    },
    "updated_at":"2019-10-28T19:37:44.361Z",
    "updated_by":{
        "href":"/users/1"
    }
}

```

```
    }  
  }  
]
```

Curl Example for POST

```
curl --request POST --url https://pce.my-company.com:8443/api/v2/orgs/1/container_
clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_profiles --header
'authorization: Basic
A5ZmRjNjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMDAzZmMzOTQ1ZGMxYzEwZGJhZTg5Nzlm
ZjM=' --header 'content-type: application/json' --data '{"name":
"test","description": "test","assign_labels": [{"href":
"/orgs/1/labels/1"}],"mode": "enforced","log_traffic": true}'
```

Curl Example for PUT

```
curl --request PUT --url https://pce.my-company.com:8443/api/v2/orgs/1/container_
clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/container_workload_
profiles/219b49c3-3bb5-4fc0-9913-b76398105e35 --header 'authorization: Basic
mRjNjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMDAzZmMzOTQ1ZGMxYzEwZGJhZTg5NzlmZjM
=' --header 'content-type: application/json' --data '{"name":
"test","description": "test","assign_labels": [{"href":
"/orgs/1/labels/1"}],"mode": "enforced","log_traffic": true}'
```

Example Response for POST

```
[  
  {  
    "href": "/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/
container_workload_profiles/219b49c3-3bb5-4fc0-9913-b76398105e35",  
    "name": "test",  
    "namespace": null,  
    "description": "test",  
    "mode": "enforced",  
    "log_traffic": true,  
    "visibility_level": "flow_summary",  
    "assign_labels": [  
      {  
        "href": "/orgs/1/labels/1",  
        "name": "test",  
        "namespace": null,  
        "description": "test",  
        "mode": "enforced",  
        "log_traffic": true,  
        "visibility_level": "flow_summary",  
        "assign_labels": []  
      }  
    ]  
  }  
]
```

```
        "href": "/orgs/1/labels/1"
      }
    ],
    "linked": false,
    "created_at": "2019-10-28T22:58:50.465Z",
    "created_by": {
      "href": "/users/1"
    },
    "updated_at": "2019-10-28T22:58:50.465Z",
    "updated_by": {
      "href": "/users/1"
    },
    "pairing_key":
    "f76079c23fbd85d4aa198814af9c09e9343c7ba24b05d6b4f86aeeb273e2b8a4"
  }
}
```

Label Restrictions

Kubernetes pods and services running in a namespace (Kubernetes) or project (OpenShift) must be labeled (RAEL) to be included in policy within Illumio Core. The container workload profile defines how labels will be assigned to pods and services within a namespace.

Illumio labels can be statically assigned from the PCE or defined in the Kubernetes manifest files using annotations. For each label key (RAEL), the PCE administrator can define four options:

1. No label will be assigned.
2. One label will be assigned from PCE.
3. A restricted list of labels can be assigned from Kubernetes using annotations. Label restrictions prevent misuse of Illumio labels by the people managing the Kubernetes platform and makes sure the labels inherit the policy they should be receiving.
4. Any label can be assigned from Kubernetes.

You can set role labels for the following APIs:

- PUT /api/v2/orgs/:xorg_id/container_clusters/<:cluster_id>/container_workload_profiles

- POST /api/v2/orgs/:xorg_id/container_clusters/<:cluster_id>/container_workload_profiles

Examples

Set an empty Role label

```
{
  "labels": [
    {"key": "role", "assign": {}} ]
}
```

Set a Location label

```
PUT /api/v2/orgs/1/container_clusters/65d1f197-938a-49ef-9343-6f55ec76fd90/container_workload_profiles/afe4661a-03ef-462f-ada6-ce7334aa9704
```

```
{
  "labels": [
    { "key": "loc", "restriction": {"href": "/orgs/1/labels/221"} } ]
}
```

Set an allow list for the Environment label

Allow a list of Environment labels to be assigned using Kubernetes:

```
PUT /api/v2/orgs/1/container_clusters/65d1f197-938a-49ef-9343-6f55ec76fd90/container_workload_profiles/afe4661a-03ef-462f-ada6-ce7334aa9704

{
  "labels": [
    { "key": "env", "restriction": [{"href": "/orgs/1/labels/176"}, {"href": "/orgs/1/labels/302"}, {"href": "/orgs/1/labels/303"}] } ]
}
```

Allow any value for the Application label

```
PUT /api/v2/orgs/1/container_clusters/65d1f197-938a-49ef-9343-6f55ec76fd90/container_workload_profiles/afe4661a-03ef-462f-ada6-ce7334aa9704

{
  "labels": [
    { "key": "app", "restriction": [] }
  ]
}
```

Multiple ways to assign or allow labels used together in one Container Workload Profile

```
PUT /api/v2/orgs/1/container_clusters/65d1f197-938a-49ef-9343-6f55ec76fd90/container_workload_profiles/afe4661a-03ef-462f-ada6-ce7334aa9704

{
  "labels": [
    { "key": "role", "assign": {} },
    { "key": "app", "restriction": [] },
    { "key": "env", "restriction": [{"href": "/orgs/1/labels/176"}, {"href": "/orgs/1/labels/302"}, {"href": "/orgs/1/labels/303"}] },
    { "key": "loc", "assign": {"href": "/orgs/1/labels/221"} }
  ]
}
```

Result for the above example:

- `role`: No label will be set; it is an explicit statement (you don't want a `role` label to be assigned).
- `app`: Any value can be set in the annotations for the `app` label key (provided the value exists in PCE).
- `env`: Only the values specified in the allowlist can be set in the annotations for the `env` label key.
- `loc`: The value of the `loc` label key is assigned to the value defined in the payload.

Clear the label assignment configuration

To clear the label assignment option and go back to the default option (any labels passed at runtime using Kubernetes annotations will be allowed), 2 options:

Option 1: explicit statement

```
{
  "labels": [
    { "key": "role", "restriction": [] }
  ]
}
```

Option 2: empty payload

```
{
  "labels": []
}
```

Container Cluster Service Backend API

Kubernetes services are represented as virtual services in the Illumio policy model. For the services in Kubernetes, Kubelink creates virtual services in the PCE and reports the list of Replication Controllers, DaemonSets, and ReplicaSets responsible for managing the pods that support the services.

When there is a match between the Replication Controller and ReplicaSet managing a pod, the PCE creates a binding between the virtual service and the container workload.

The Service Backend represents a match between a virtual service and an application type, such as Deployment or ReplicaSet.

Use this method to get the service backend:

Functionality	HTTP	URI
Get data about the service backend	GET	GET /orgs/1/container_clusters/:container_cluster_id/service_backends

Query Parameters

Parameters	Description	Type	Required
name	The name of the container cluster backend.	String	Yes

Parameters	Description	Type	Required
kind	The type (or kind) of the container cluster backend.	String	Yes
namespace	The namespace of the container cluster backend.	String	No
updated_at	The time (rfc339 timestamp) at which the container cluster backend was updated.	String	Yes
created_at	The time (rfc339 timestamp) at which the container cluster backend was created.	String	Yes
virtual_services	Includes the following properties: <ul style="list-style-type: none"> href: The URI to the associated virtual service name: The virtual service name 	Object String String	Yes

Curl Examples

Curl Example for GET

```
curl --request GET --url https://pce.my-company.com:8443/api/v2/orgs/1/container_clusters/445bfa9b-4de4-4c09-9705-496eb04b190f/service_backends --header 'authorization: Basic YzE2YjBkYjI0MjJhZGNkYWU5OjA5ZmRjNjA4MDhiMzExZTc2Y2UyNzNmOWNiN2ZhMTA5OTdkMWNlMDAzMmZ0TQ1ZGMxYzEwZGJhZTg5NzlmZjM='
```

Example Response for GET

```
[
  {
    "name": "58687784f9",
    "kind": "replicasethash",
    "namespace": "kube-system",
    "updated_at": "2019-10-25T20:07:39.741Z",
    "created_at": "2019-10-25T20:07:39.741Z",
    "virtual_service": {
      "href": "/orgs/1/sec_policy/draft/virtual_services/926c2f63-bcd8-42f1-8811-165b34f84334",
      "name": "coredns-k8s2-kube-system"
    }
  },
  {
```

```
    "name": "556b9ff8f8",
    "kind": "replicasethash",
    "namespace": "kube-system",
    "updated_at": "2019-10-25T20:07:39.768Z",
    "created_at": "2019-10-25T20:07:39.768Z",
    "virtual_service": {
      "href": "/orgs/1/sec_policy/draft/virtual_services/58b0df03-1151-464e-8352-069e3ad0d7ed",
      "name": "kubernetes-dashboard-k8s2-kube-system"
    }
  }
]
```

Access Restrictions and Trusted Proxy IPs

To employ automation for managing the PCE environment, you can use API Keys created by an admin user and automate the PCE management tasks. Illumio provides a way to restrict the usage of these API keys by IP address so that you can block API requests coming in from non-allowed IP addresses.

Access Restrictions

Access restrictions are configurable entities and contain a list of up to 8 IPv4 IP addresses or CIDR blocks that specify the source IP addresses of the allowed clients. Only the global Org Owner can manage access restrictions in the organization while other roles can neither edit nor view them.

The following rules apply to access restrictions:

- Each access restriction can be applied to either one or both:
 - API requests authenticated by API keys
 - API requests authenticated by Username/Password credentials
- The global Org Owners can edit an access restriction after it has been created by modifying the allowed IP list or the options. They can also assign access restrictions to Local and External Users. The API supports the update of access restrictions for a list of users.
- Access restrictions are leader-owned configuration objects that are replicated to all supercluster regions.

- Access restrictions are enforced as follows:
 - To enforce an API request, determine the user account for that API request using the API key or the user session token and then find the access restriction that is configured for that user. If there is no access restriction assigned to the user, the API request proceeds.
 - If the client IP address for an API request does not satisfy the corresponding user's access restrictions, the request is rejected with a 401 error message.
 - Access restrictions are not enforced on some URLs (node_available, static JS/CSS content).
- When a request is rejected due to unsatisfied access restrictions, it generates an Event that specifies a failure caused by an invalid source IP address, including the actual IP address and an appropriate error code (403).

Access Restrictions Assignment to Users

Each Access Restriction is a configuration object that specifies a set of allow-list IP addresses or CIDR blocks, designating the allowed client IP address. It also specifies the types of API accesses that are restricted (those authenticated by API Keys or those authenticated by user session tokens).

The Org Owners create and manage access restrictions in their organizations so that there are maximum of 50 access restrictions per organization. The Org Owners can assign a single access restriction to each Local or External User (by default, a user has no access restriction assigned).

Access Restriction Methods

Functionality	HTTP	URI
Get a list of access restrictions	GET	/api/v2/orgs/<org_id>/access_restrictions
Get a specific access restriction	Get	/api/v2/orgs/<org_id>/access_restrictions/<id>
Create an access restriction	POST	/api/v2/orgs/<org_id>/access_restrictions
Update an access restriction. Same schema as for POST, but fields such as name or ips might not be required	PUT	/api/v2/orgs/<org_id>/access_restrictions/<id>
The DELETE endpoint should return an error if the specified access_restriction is referenced	DELETE	/api/v2/orgs/<org_id>/access_restrictions/<id>

Functionality	HTTP	URI
by any User or Group. The existing access_restrictions from all Users and Groups must be removed before they can be deleted.		

Return Values for Access Restriction

These are the return values for the Access Restriction methods:

Parameter	Method	Description	Required
href	GET	URI of access restriction	Yes
name	GET, POST,	User assigned name of the access restriction	(No GET) (Yes POST)
description	GET, POST	User assigned description of the access restriction	No
ips	GET, POST	Array of ip addresses or CIDR blocks	Yes
enforcement_exclusions	GET, POST	The types of API access methods that are excluded from access restriction enforcement	No

Curl Commands to Manage Access Restrictions

Create an Access Restrictions

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/1/access_restrictions/
```

Response

```
{
  "name": "sample Access Restriction payload",
  "description": "example",
  "ips": [ "192.168.33.1/16" ],
  "enforcement_exclusions": [ "user_sessions" ]
}
```

Read an Access Restriction

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/1/access_restrictions/
```

Update an Access Restriction

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/1/access_restrictions/1
```

```
{
  "name": "modified Access Restriction payload",
  "description": "example",
  "ips": [ "192.168.33.1/16" ],
  "enforcement_exclusions": [ "user_sessions" ]
}
```

Delete the Access Restriction

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/1/access_
restrictions/1
```

Curl Command to associate an Access Restriction with an Org Auth Sec Principal (PUT)

```
curl -i -X -PUT https://pce.my-company.com:8443/api/v2/orgs/1/auth_security_
principals/76a0607b-6961-4c74-a98a-8b10775c8a9b
```

```
{
  "name": "test.user@illumio.com",
  "display_name": "test",
  "type": "user",
  "access_restriction": {
    "href": "/orgs/1/access_restrictions/1"
  }
}
```

Trusted Proxy IPs

When a client is connected to the PCEs haproxy server, this connection can traverse one or more load balancers or proxies. Therefore, the source IP address of a client connection to haproxy might not be the actual public IP address of the client.

Proxies and intermediaries often use the X-Forwarded-For header (and some other custom headers, like X-Client-IP) to pass along the client IP address. The value of this

header is a comma-separated list of one or more IP addresses, where the source IP address seen by the most recent proxy is at the end of the list.

The client IP address used for API requests and Web UI connections comes from the value of the X-Forwarded-For header that haproxy sets on the back-end request to the webservice. It is set to the one of these values:

- Value of the X-Forwarded-For header on the incoming request (when trust_upstream_x_forwarded_for is true)
- Source IP address of the client connection to haproxy (when trust_upstream_x_forwarded_for is false)

Configurable trusted proxy IPs allow Org Owners to configure a list of IPv4 addresses or CIDR blocks that are considered trusted for setting a client's X-Forwarded-For header. Using this setting, the Org Owner can designate the trusted proxies/intermediaries and the PCE will consider all others to be un-trusted for the purpose of setting the X-Forwarded-For header.

The haproxy is configured to always put the client's source IP address in the X-Real-IP header on the back-end request and to pass along any X-Forwarded-For headers that are in the front-end request.

Trusted Proxy IP Methods

Functionality	HTTP	URI
Get a list of trusted IP proxies	GET	/api/v2/orgs/<org_id>/-settings/trusted_proxy_ips
Interservice API for fetching an orgs' trusted_proxy_ips settings, so that it may be cached locally. It uses the same schema as the GET endpoint above; it receives the org_id as a quer input	GET	/api/v2/org_trusted_proxy_ips?org_id=<id>
Update trusted_proxy_ips settings for a given org, with the same schema as the GET endpoint (except without the max_trusted_proxy_ips_per_region property)	PUT	/api/v2/orgs/<org_id>/-settings/trusted_proxy_ips

Return Values for Trusted Proxy IPs

These are the return values for the Trusted Proxy methods:

Parameter	Method	Description	Required
max_trusted_proxy_ips_per_region	GET	Maximum number of Trusted Proxy IPs allowed for each PCE	Yes
trusted_proxy_ips	GET, PUT	IPs or CIDRs trusted (per-region) for handling clients' X-Forwarded-For header Required: pce_fqdn: FQDN of PCE region, or null if not in supercluster ip: IP address or CIDR trusted for handling clients' X-Forwarded-For header	Yes

Curl Commands to Manage Trusted Proxy IPs

Read a Trusted Proxy IP

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/1/access_restrictions/
```

Update a Trusted Proxy IP

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/1/settings/trusted_proxy_ips/
```

```
{
  "trusted_proxy_ips": [
    {
      "pce_fqdn": null,
      "ip": "66.151.147.0/24"
    },
    {
      "pce_fqdn": null,
      "ip": "192.168.34.0/24"
    }
  ]
}
```

Provisioning

This chapter contains the following topics:

Policy Update Mode	131
Provisioning (public stable)	136
Provisioning (public experimental)	141
Selective Enforcement	152
Virtual Server Filtering	158

Use the Public Stable Provisioning API to implement all current changes to your security policy, such as additions, changes, and deletions.

The Public Experimental Provisioning API supplies information about unprovisioned changes to security policy items.

Finally, the Policy Update Mode API controls when policy updates are applied to workloads.

Policy Update Mode

This Public Experimental API controls when policy updates are applied to workloads.

Overview of Policy Update Mode

The PCE has two policy update options:

- **Adaptive:** Apply policy changes as soon as you provision.
- **Static:** Apply policy changes at a later time, such as during a scheduled maintenance window.

By default, the PCE policy update mode is set to Adaptive, but you can configure Static policy update mode for certain sets of workloads identified by scopes. Workloads that share the same labels configured for static policy update scope *receive* policy changes from the PCE, but those changes *will not be applied* until a user or an orchestration system instructs the PCE to apply those changes.

Configuring static policy update mode requires defining a scope that contains one or more environment, application, or location labels (scopes do not have role labels). If a label type is not defined in the scope, that label type is interpreted as All. For example, if the policy update scope is

Application = Checking, Location = China,
the PCE interprets the scope as

Application = Checking, Location = China, Environment = All.

Policy Update Mode Methods

Functionality	HTTP	URI
Get the current policy update mode for your organization	GET	[api_version][org_href]/sec_policy/:pversion/firewall_settings
Change the policy update mode for your organization	PUT	[api_version][org_href]/sec_policy/:pversion/firewall_settings

Get Policy Update Mode

You can use this method to get the current policy update mode settings for your organization, which is part of your PCE security settings. This method contains a variable (:pversion) that can be used to return the security settings with active (currently provisioned) or draft state for your organization.

URI To Get Policy Update Mode

```
GET [api_version][org_href]/sec_policy/:pversion/firewall_settings
```

Draft or Active Policy Update Mode

Parameter	Description	Type
:pversion	Allows you to get: <ul style="list-style-type: none"> active: The currently provisioned security settings, including policy update mode 	String

Parameter	Description	Type
	<ul style="list-style-type: none">draft: The draft state of any changed security settings that have not yet been provisioned, including policy update mode	

Curl Command Get Active Policy Update Mode

This curl example gets the active (currently provisioned) security settings for your organization, which includes the policy update mode settings.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/sec_policy/active/firewall_settings -H "Accept: application/json" -u $KEY:$TOKEN
```

Response Body

The `static_policy_scopes` property in the response (in **blue**) indicates that two static scopes have been configured for policy update.

Each scope is defined as a JSON array of labels, which includes an Application, Environment, and a Location label. Labels in the scope are identified by their HREFs.

```
{
  "href": "/orgs/7/sec_policy/active/firewall_settings",
  "created_at": "2015-10-23T22:01:01.151Z",
  "updated_at": "2017-09-02T19:08:55.623Z",
  "deleted_at": null,
  "created_by": { "href": "/users/0" },
  "updated_by": { "href": "/users/14" },
  "deleted_by": null,
  "update_type": null,
  "allow_dhcp_client": true,
  "log_dropped_multicast": true,
  "log_dropped_broadcast": false,
  "allow_traceroute": true,
  "allow_ipv6": true,
  "allow_igmp": false,
  "track_flow": true,
  "system_rule_log_flow": false,
  "allow_path_mtu_discovery": true,
  "network_detection_mode": "single_private_brn",
  "static_policy_scopes": [
```

```

[
  { "label": { "href": "/orgs/7/labels/83" } },
  { "label": { "href": "/orgs/7/labels/86" } },
  { "label": { "href": "/orgs/7/labels/94" } }
],
[
  { "label": { "href": "/orgs/7/labels/82" } },
  { "label": { "href": "/orgs/7/labels/100" } },
  { "label": { "href": "/orgs/7/labels/89" } },
  { "label": { "href": "/orgs/7/labels/94" } }
]
],
"secure_connect_certs": {
  "default_issuer_name_match": "test",
  "scoped_certificates": []
}
}

```

Change Policy Update Mode

Sets your organization's draft policy update mode, which might include adding or removing a policy scope.

The draft state of your policy update mode can be modified, but not the currently active (provisioned) version. First, change to the draft policy update mode, and then provision those changes.

URI To Change Policy Update Mode

```
PUT [api_version][org_href]/sec_policy/:pversion/firewall_settings
```

Request Properties

Property	Description	Type	Required
static_policy_scopes	<p>A set of up to three labels, one or more of the type Application, Environment, and Location.</p> <p>Each label in the policy scope is identified by its HREF nested in a JSON array.</p> <p>Before you update the organization policy update mode,</p>	JSON array of strings	Yes

Property	Description	Type	Required
	make sure that you have the exact set of labels you want to use and their HREFs.		

Request Body

This example shows the request body for two policy update scopes. The first has a single label scope, and the second scope has a set of three labels.

```
{
  "static_policy_scopes": [
    [
      { "label": { "href": "/orgs/1/labels/8" } }
    ],
    [
      { "label": { "href": "/orgs/1/labels/2" } },
      { "label": { "href": "/orgs/1/labels/8" } },
      { "label": { "href": "/orgs/1/labels/11" } }
    ]
  ]
}
```

Curl Command to Update Policy Update Mode

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/7/firewall_settings -H
"Content-Type: application/json" -u $KEY:$TOKEN -d '{"static_policy_scopes":
[[{"label":{"href":"/orgs/1/labels/8"}},{{"label":{"href":"/orgs/1/labels/2"}},
{"label":{"href":"/orgs/1/labels/8"}},{{"label":{"href":"/orgs/1/labels/11"}}]]}'
```

Response

The response for a successful change to your policy update mode is an HTTP 204 No Content Operation. No data is returned.

Remove all Static Policy Scopes

To remove all static policy scopes, pass an empty JSON array:

```
PUT [api_version][org_href]/sec_policy/:pversion/firewall_settings { "static_
policy_scopes": [] }
```

Curl Command to Remove Static Policy Scopes

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/7/firewall_settings -H
"Content-Type: application/json" -u $KEY:$TOKEN -d '{"static_policy_scopes":[]}'
```

Provisioning (public stable)

This Public Stable API provisions all current changes (additions, changes, and deletions) to your security policy.

This API can also return a collection of provisioning versions or an individual provisioning version.

To get information about unprovisioned changes to security policy items, find provisioning dependencies, delete unprovisioned security policy items, revert the last provisioned items, and check whether a security rules exists that allows communications between two workloads, see [Provisioning \(public experimental\)](#).

Provisioning API Methods

Functionality	HTTP	URI
Provision the current set of modified security policy items	POST	[api_version][org_href]/sec_policy
Get a list of all provisioned security policy versions	GET	[api_version][org_href]/sec_policy
Get a specific version of a provisioned security policy	GET	[api_version][sec_policy_version_href]

Provision All Items

Policy item additions, modifications, and deletions must be provisioned before they take effect on workloads.

URI to Provision All Items

```
POST [api_version][org_href]/sec_policy
```


Curl Command to Provision All Items

**NOTE:**

This example passes a provisioning comment using the curl -d option (lowercase d) followed by the comment '{"commit_message":"make active"}'. This operation provisions all draft policy items.

**NOTE:**

For v2 the request body for provision all would be -d '{"update_description":"make active"}'

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy -H "Content-Type: application/json" -u $KEY:$TOKEN -d '{"commit_message":"make active"}'
```

Response

After provisioning the draft security policy, the response provides information related to the operation, including the version HREF of the provisioning.

You can use a provision history HREF to get all modified items for a particular version.

The response also indicates how many workloads were affected, when the provisioning was done, which user did it, and any message that was provided.

```
{
  "href": "/orgs/2/sec_policy/80",
  "commit_message": null,
  "version": 80,
  "workloads_affected": 3,
  "created_at": "2015-09-26T21:48:46.446Z",
  "created_by": { "href": "/users/18" }
}
```

Provision Individual Items

Curl Example

The request body uses update_description instead of commit_message, and instead of entities, define an array of pending HREFs for each method as appropriate.

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"change_subset":
{"rule_sets":[{"href": "/orgs/2/sec_policy/draft/rule_sets/843"}], "ip_lists":
[{"href": "/orgs/2/sec_policy/draft/ip_lists/151"}], "update_
description":"Provisioning a ruleset and an ip list"}'
```

Request Body Prototype

The security policy POST request body has this format. Only define the methods used in the call and don't include any unused methods in the request body.

```
{
  "update_description": "string",
  "change_subset": {
    "label_groups": [
      {
        "href": "string"
      }
    ],
    "services": [
      {
        "href": "string"
      }
    ],
    "rule_sets": [
      {
        "href": "string"
      }
    ],
    "ip_lists": [
      {
        "href": "string"
      }
    ],
    "virtual_services": [
      {
        "href": "string"
      }
    ]
  },
  ]
}
```

```
"firewall_settings": [  
  {  
    "href": "string"  
  }  
],  
"secure_connect_gateways": [  
  {  
    "href": "string"  
  }  
],  
"virtual_servers": [  
  {  
    "href": "string"  
  }  
]  
}
```

Restore the Previous Security Policy

This API restores the previous security policy. The JSON request body is an empty set of curly braces.

Curl Command to Restore the Security Policy

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy -H  
"Content-Type: application/json" -u $KEY:$TOKEN -d {}
```

Get All Provision Versions

This method gets the full history of all provisioned security policy versions.

URI to Get All Provisioned Versions

```
GET [api_version][org_href]/sec_policy
```

Curl Command to Get the Provision Versions

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

```
{
  "href": "/orgs/2/sec_policy/80",
  "commit_message": null,
  "version": 80,
  "workloads_affected": 3,
  "created_at": "2015-09-26T21:48:46.446Z",
  "created_by": { "href": "/users/18" }
},
{
  "href": "/orgs/2/sec_policy/79",
  "commit_message": "",
  "version": 79,
  "workloads_affected": 2,
  "created_at": "2015-09-24T19:40:31.340Z",
  "created_by": { "href": "/users/18" }
},
{
  "href": "/orgs/2/sec_policy/78",
  "commit_message": "",
  "version": 78,
  "workloads_affected": null,
  "created_at": "2015-09-24T19:21:03.484Z",
  "created_by": { "href": "/users/18" }
}
```

Get an Individual Provision Version

This method gets a specific version of a provisioned policy.

Every time security policy is provisioned, it gets a unique version ID, which takes the form of an HREF. This HREF can be obtained from a GET of all security policy provisioned versions and then used in this call.

URI to Get an Individual Version of a Provisioned Policy

```
GET [api_version][sec_policy_version_href]
```

Curl Command to Get Version

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/79 -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

```
{
  "href": "/orgs/2/sec_policy/79",
  "commit_message": "",
  "version": 79,
  "workloads_affected": 2,
  "created_at": "2015-09-24T19:40:31.340Z",
  "created_by": { "href": "/users/18" }
}
```

Provisioning (public experimental)

This Public Experimental API gets information about unprovisioned changes to security policy items (rulesets, IP lists, security settings, labels and label groups, services, virtual services, and user groups). You can also find provisioning dependencies, delete unprovisioned security policy items, revert the last provisioned items, and check whether a security rule exists that allows communications between two workloads.

To provision security policy items and get information about one or more provisioned items, see [Provisioning \(public stable\)](#).

Provisioning API Methods

Functionality	HTTP	URI
Get the collection of modified (draft) security policy items pending provisioning.	GET	[api_version][org_href]/sec_policy/pending
Delete all unprovisioned security policy item modifications (all unprovisioned draft changes) pending provisioning.	DELETE	[api_version][org_href]/sec_policy/pending
Revert a specified list of pending uncommitted security policy items. This method allows you to select specific items to revert.	PUT	[api_version][org_href]/sec_policy/delete
Determine if a specific set of objects can be provisioned, or if they are dependent on other objects that need to be provisioned as well.	POST	[api_version]/sec_policy/draft/dependencies
Get the collection of all policy items that were modified in a specific version of a security policy.	GET	[api_version][sec_policy_version_href]/modified_objects
Check whether a rule exists between two workloads that allows communication.	GET	[api_version][sec_policy_version_href]/allow

Provisionable Policy Items

The following security policy items all require provisioning before they can take effect on managed workloads (workloads with a VEN installed on them). The total sum of these policy items constitutes the security policy.

- **IP Lists:** IP addresses, IP ranges, and CIDR blocks allowed to access managed workloads.
- **Label Groups:** Labels can be managed in label groups.
- **Rulesets:** Policy item that includes labels and rules to define permitted communication between workloads and between groups.
- **Pairing Profiles:** A pairing profile applies certain properties to workloads as they pair with the PCE, such as labels and workload policy states.
- **Security Settings:** General network security settings, such as ICMP echo reply, allow or disable IPv6, and connectivity settings.

- **Services:** Definitions or discovery of existing services on your workloads.
- **Virtual Servers:** Allows rules that allow communication with workloads managed by a load balancer.
- **Virtual Services:** A virtual service is a single service (a port/protocol set) that can be used directly in a rule as a single entity. Labels that represent multiple virtual services can also be used to write rules.

When the security policy is provisioned, the PCE recalculates any changes made to policy configurations and then transmits those changes to the VENs installed on the workloads.

Policy Provisioning States

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, fire-wall settings, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID of the security policy.

Get All Items Pending Provisioning

This method gets a list of all modified policy items pending provisioning.

URI to Get All Policy Items Pending Provisioning

```
GET [api_version][org_href]/sec_policy/pending
```

Curl Command to Get Items Pending Provisioning

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/pending -H "Accept:application/json" -u $KEY:$TOKEN
```

Response

```
{
  "rule_sets": [
    {
      "href": "/orgs/1/sec_policy/draft/rule_sets/598",
      "updated_by": {
        "href": "/users/xx"
      },
      "updated_at": "2018-10-13T04:xx:40.317Z",
      "update_type": "update",
      "caps": [
        "write",
        "provision"
      ],
      "name": "network-services@xxxx503325.xxxx | en44 | eastern"
    }
  ]
}
```

Revert All Items Pending Provisioning

This method reverts (undoes) the current set of unprovisioned security policy modifications (all unprovisioned draft changes).

Revert a List of Items Pending Provisioning

This method reverts individual unprovisioned security policy items that have been added, modified, or deleted.

Before using this method, identify the individual security policy items to revert by including the unique policy item HREF.

URI to Revert a Specific List of Items Pending Provisioning

```
PUT [api_version][org_href]/sec_policy/delete
```

Request Body

Add the HREF in the request body for each security policy item to revert.

In this example, the request body contains an unprovisioned modified ruleset to revert identified by its HREF.


```
{
  "entities": {"href":"/orgs/2/sec_policy/draft/rule_sets/12634/sec_rules/5454"}
}
```

Curl Command to Revert a Pending Rule

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/1/sec_policy/delete -H
"Accept: application/json" -H "Content-Type: application/json" -u api_
1fc24761346777702:'26c55be6892762b65f27aacc795076767f16ffcd7e9fde323a307e5fd286eb8
d' -d '{"change_subset":{"rule_sets":[{"href":"/orgs/1/sec_policy/draft/rule_
sets/3"}]}}'
```

Get Provisioning Dependencies

This method determines if a specific set of objects can be provisioned, or if they are dependent on other objects that also must be provisioned. This method can also determine if any changes to revert have dependencies before you revert them.

URI to Get Specific Provisioning Dependencies



NOTE:

This method must have draft in the URI because it only checks provisioning or reverting dependencies for objects that have not yet been provisioned.

```
POST [api_version]/sec_policy/draft/dependencies
```

Request Body Properties

Parameter	Description	Type	Req
change_subset	Defines a hash of provisionable or revertible objects identified by their HREFs. Includes label groups, services, rulesets, IP lists, virtual services, and virtual servers. Each individual object of a specific type. For example, rule_sets is represented in the request body as an array of HREFs for those object types.	Object	Yes

Parameter	Description	Type	Req
operation	<p>Determines if there are dependencies for <i>provisioning</i> or <i>reverting</i> the specified objects:</p> <ul style="list-style-type: none"> • <code>commit</code>: Specify this value to check for dependencies before <i>provisioning</i> an object. • <code>revert</code>: Specify this value to check for dependencies before <i>reverting</i> an object that is in a draft state. 	String	Yes
Sub properties of <code>change_subset</code> that represent provisionable objects		Array	No
label_groups	List of label groups in the draft state to check for provisioning dependencies identified by label group HREF.	String	No
services	List of services in the draft state to check for provisioning dependencies identified by service HREF.	String.	No
rule_sets	List of rulesets in the draft state to check for provisioning dependencies identified by rule_set HREF.	String	No
ip_lists	List of IP lists in the draft state to check for provisioning dependencies, identified by IP list HREF.	String	No
virtual_services	List of virtual services in the draft state to check for provisioning dependencies identified by virtual service HREF.	String	No
virtual_servers	List of virtual servers in the draft state that you want to check for provisioning dependencies identified by virtual server HREF.	String	No
firewall_settings	<p>List of security settings (<code>firewall_settings</code>) in the draft state to check for provisioning dependencies identified by the security settings HREF for your PCE.</p> <p>Currently, the only security setting that can be</p>	String	No

Parameter	Description	Type	Req
	modified with the REST API is the policy update mode. For more information, see Policy Update Mode .		

Request Body (v2)

This example determines if two rulesets and two virtual services that have been modified have any dependencies before the changes are provisioned. Each ruleset and virtual service is identified by its HREF, which can be obtained from the response from a GET collection of those objects.

```
{
  "change_subset": {
    "rule_sets": [
      {
        "href": "/orgs/1/sec_policy/draft/rule_sets/9"
      },
      {
        "href": "/orgs/1/sec_policy/draft/rule_sets/3"
      }
    ],
    "virtual_services": [
      {
        "href": "/orgs/1/sec_policy/draft/virtual_services/xxxxxxx-adeb-4895-8ff2-60c5b9833d9e"
      },
      {
        "href": "/orgs/1/sec_policy/draft/virtual_services/xxxxxxx-12bc-4cfa-99ef-330c399bc78c"
      }
    ]
  },
  "operation": "commit"
}
```

Curl Command to Check for Provisioning Dependencies

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/7/sec_policy/draft/dependencies -H "Content-Type: application/json" -u $KEY:$TOKEN -d '
```

```
{ "operation": "commit", "change_subset": { "rule_sets": [ { "href": "/orgs/1/sec_policy/draft/rule_sets/9" }, { "href": "/orgs/1/sec_policy/draft/rule_sets/3" } ], "virtual_services": [ { "href": "/orgs/1/sec_policy/draft/virtual_services/xxxxxxxx-adeb-4895-8ff2-60c5b9833d9e" }, { "href": "/orgs/1/sec_policy/draft/virtual_services/xxxxxxxx-12bc-4cfa-99ef-330c399bc78c" } ] } }
```

Response

The response indicates if these objects have dependencies.

```
[
  {
    "dependency": {
      "services": {
        "href": "/orgs/1/sec_policy/draft/services/5",
        "caps": [
          "provision"
        ]
      }
    },
    "required_by": {
      "virtual_services": [
        {
          "href": "/orgs/1/sec_policy/draft/virtual_services/b9faf240-adeb-4895-8ff2-60c5b9833d9e",
          "caps": [
            "provision"
          ]
        }
      ],
      "rule_sets": [
        {
          "href": "/orgs/1/sec_policy/draft/rule_sets/9",
          "caps": [
            "write",
            "provision"
          ]
        }
      ]
    }
  ]
}
```

```
    }
  },
  {
    "dependency": {
      "ip_lists": {
        "href": "/orgs/1/sec_policy/draft/ip_lists/5",
        "caps": [
          "provision"
        ]
      }
    },
    "required_by": {
      "rule_sets": [
        {
          "href": "/orgs/1/sec_policy/draft/rule_sets/3",
          "caps": [
            "write",
            "provision"
          ]
        }
      ]
    }
  }
]
```

If there are no dependencies for either commit or revert, the response returns an empty array.

```
[]
```

Get Modified Items in a Provisioned Version

This method gets a collection of all modified policy items in a specific version of the security policy.

Every time the security policy is provisioned, it gets a version, which takes the form of an HREF. The HREF can be obtained when getting all provisioned versions of your security policy. You can use that provision version HREF when calling this method.

URI to Get All Modified Items in a Specific Provisioned Version

```
GET [api_version][sec_policy_version_href]/modified_objects
```

Curl Command Example

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/19/modified_objects -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

```
[
  {
    "object_type": "rule_sets",
    "update_type": "deleted",
    "href": "/orgs/1/sec_policy/xxx/rule_sets/xxx",
    "updated_at": "2018-08-xxTxx:37:52.974Z",
    "updated_by": {
      "href": "/users/xx"
    },
    "name": "3tier-applications@xxxxx862852.xxxxx | prod"
  }
]
```

Get Rules Allowing Communication

This method gets a list of all rules that allow communication between two workloads (and other entities) for a specific version of a provisioned security policy.

By default, the maximum number returned on a GET collection with this API is 500. If you want to get more than 500 results, use an [Asynchronous GET Collection](#).

URI to Check for Rules Between Workloads

```
GET [api_version][sec_policy_version_href]/allow
```

Query Parameters

Provide query parameters in the URI that specify the source workload IP address or HREF, the service HREF, and the destination workload HREF. You can obtain a workload HREF with a [GET call on the Workloads API](#).

Parameter	Description	Type
src_external_ip OR src_workload	Indicates the source workload by either the workload IP address or the workload URI, but not both.	String
dst_external_ip OR dst_workload	Indicates the destination workload by either the workload IP address or the workload URI, but not both.	String
service	The service HREF.	String
port	The port used by the service.	Integer
protocol	The protocol used by the service.	Integer

Curl Command to Get Rules Between Workloads

The workloads and the service are identified by their HREFs:

```
curl -i -X GET -H "Accept: application/json" -u $KEY:$TOKEN https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/draft/allow?src_workload=/orgs/2/workloads/xxxxxxx-aa40-4461-8738-80a30c667e08&dst_workload=/orgs/2/workloads/xxxxxxx-4d31-4b1b-af69-60eb134b06b6&service=/orgs/2/sec_policy/draft/services/1&port=80&protocol=6
```

Response

```
{
  "href": "/orgs/2/sec_policy/draft/rule_sets/224/sec_rules/302",
  "created_at": "2015-09-22T22:58:50.787Z",
  "updated_at": "2015-09-22T22:58:50.787Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/6"
  },
  "updated_by": {
    "href": "/users/6"
  },
  "deleted_by": null,
  "description": null,
}
```

```
"enabled": true,
"sec_connect": true,
"providers": [
  {
    "label": {
      "href": "/orgs/2/labels/133"
    }
  }
],
"consumers": [
  {
    "label": {
      "href": "/orgs/2/labels/133"
    }
  }
],
"ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/620"}],
"resolve_labels_as": {
  "providers": ["workloads"],
  "consumers": ["workloads"]
}
}
```

Selective Enforcement

Selective enforcement represents an important improvement to enable you to protect only a subset of applications or processes on a workload. While the selected applications or processes are protected, the other services or ports behave as if the whole workload is "in visibility" or `visibility_only` (formerly known as the "build" mode). When using selective enforcement, you can also temporarily enforce specific ports in cases when a vulnerability is detected and swift action is needed.

Selective enforcement, therefore, helps overcome the limitations of the allow-list security model that requires a complete list of what's allowed in order to enforce any security. It enables security enforcement before the customer can build a complete allow-list.

The important use cases for selective enforcement are:

- Emergency risk reduction: If you detect a vulnerability that you cannot easily patch, use selective enforcement, and quickly lock the affected ports without changing the behavior of other ports or applications.
- Selective enforcement of a broader set of ports: Start by enforcing only the most critical ports (such as port 22) and gradually add more ports until you reach full enforcement.

Various RBAC roles (such as Workload Manager, Ruleset Manager, Ruleset Viewer, Ruleset Provisioner, Global Read-Only user, or Global Object Provisioner) have their specific privileges regarding selective enforcement policies, while only the Global Org Owner or Global Admin can manage enforcement policy with no restrictions.

The enforcement policies follow the allow-list model. This means that enforcement policies are additive and, therefore, will never contradict themselves.

Selective Enforcement applies only to managed workloads; it does not apply to NEN-controlled or other unmanaged workloads.

Selective enforcement applies only to inbound connections.

Users can view, create, edit, and delete selective enforcement policies using the new Selective Enforcement API methods.

How Does Selective Enforcement work

Selective Enforcement enforces one or more ports on a workload (named. "port enforcement") while leaving the remaining ports unenforced. It controls enforcement using policies (scopes) instead of configuring workloads directly.

To use Selective Enforcement, you need to perform two separate configurations:

- Set the Workload policy state to `Selective Enforcement`.
- Create a Selective Enforcement policy with a scope that includes the workload.

Changes introduced with Selective Enforcement are:

- In workload configuration: Traffic Logging and Policy State are now two separate settings.
- In workload configuration: A new Selective Enforcement policy state is introduced, as well as a new Selective Enforcement Policy Object.

Selective Enforcement Methods

Functionality	HTTP	URI
Allow users to view the configured selective enforcement rules.	GET	[api_version][org_href]/sec_policy/:version/selective_enforcement_rules

Functionality	HTTP	URI
Edit the configured selective enforcement rules	PUT	[api_version][org_href]/sec_policy/:version/selective_enforcement_rules/:id
Create a new selective enforcement rule.	POST	[api_version][org_href]/sec_policy/:version/selective_enforcement_rules/:id
Delete the specified selective enforcement rule.	DELETE	[api_version][org_href]/sec_policy/:version/selective_enforcement_rules/:id

To introduce the feature Selective Enforcement, the existing common schema `workload_modes` has been DEPRECATED:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "DEPRECATED AND REPLACED (Use enforcement_mode instead)",
  "type": "string",
  "enum": ["idle", "illuminated", "enforced"]
}
```

This schema has been substituted with the one named `workload_enforcement_mode`, which contains the option for selective enforcement:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "Workload enforcement mode",
  "type": "string",
  "enum": ["idle", "visibility_only", "full", "selective"]
}
```

These two schemas correlate as follows (old vs. new):

- `idle` = `idle` (no changes)
- `illuminated` (build, test) = `visibility_only` (no traffic is blocked by policy)
- `enforced` = `full` (segmentation rules are enforced for all inbound and outbound services. Traffic not allowed by the segmentation rule is blocked.)

- **selective:** New starting from the release 20.2. Segmentation rules are enforced only for the selected inbound services when workload is within the scope of the Selective Enforcement rule.

Query Parameters for Selective Enforcement

These are the query parameters for Selective Enforcement :

Parameter	Description	Type
labels	List of lists of label URIs, encoded as a JSON string	String
name	Filter by name supports partial matching	String
external_data_set	The data source from which a resource originates	String
external_data_ref- erence	A unique identifier within the external data source	String
service		String
service_ports.- proto	Protocol to filter on	Integer
service_ports.port	Specify port or port range to filter results. The range is from -1 to 65535.	String
max_results	Maximum number of Rule Sets to return	Integer

Return Values for Selective Enforcement

These are the return values for Selective Enforcement :

	Description	Type
href	(GET only) URI of the selective enforcement rule	String
created_at	(GET only) Timestamp when this rule set was first created. Format date-time	String
updated_at	(GET only) Timestamp when this rule set was last updated. Format date-time	String
created_by	(GET only) User who originally created this rule set. Required parameter href.	String
updated_by	(GET only) User who last updated this rule set. Required parameter href.	String
name	Name of the selective enforcement rule	String
scope	labelLabel URI. Required parameter is href. label_groupLabel group URI. Required parameter is href.	Array Object String Object String

	Description	Type
enforced_services	Collection of services that are enforced	Array
	port: Port number, or the starting port of a range. If unspecified, this will apply to all ports for the given protocol.	Integer
	type: integer, minimum: 0, maximum: 65535	Integer
	to_port: Upper end of port range; this field should not be included if specifying an individual port.	Integer
	proto: Transport protocol (numeric)	

Curl Command to Get Selective Enforcement Rules

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/sec_policy/draft/selective_enforcement_rules
```

Response

```
[
  {
    "href": "/orgs/1/sec_policy/draft/selective_enforcement_rules/1",
    "created_at": "2020-01-31T01:43:06.560Z",
    "updated_at": "2020-01-31T19:07:59.756Z",
    "deleted_at": null,
    "created_by": {
      "href": "/users/1"
    },
    "updated_by": {
      "href": "/users/1"
    },
    "deleted_by": null,
    "update_type": null,
    "name": "test",
    "scope": [],
  }
]
```

```
    "enforced_services": [
      {
        "href": "/orgs/1/sec_policy/draft/services/5"
      },
      {
        "port": 2456,
        "proto": 6
      }
    ],
    "caps": [
      "write",
      "provision"
    ]
  }
]
```

Curl Command to Create Selective Enforcement Rules

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/3/sec_
policy/draft/selective_enforcement_rules
```

```
{
  "scope": [
    {"label": {"href": "/orgs/1/labels/1"}},
    {"label": {"href": "/orgs/1/labels/10"}},
    {"label_group": {"href": "/orgs/1/sec_policy/active/label_groups/8c6c50fc-4eac-
11ea-b687-ef0be7369e0c"}}
  ],
  "enforced_services": [
    {
      "href": "/orgs/1/sec_policy/draft/services/5"
    },
    {
      "port": 2456,
      "proto": 6
    }
  ],
}
```

Curl Command to Edit Selective Enforcement Rules

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/3/sec_
policy/draft/selective_enforcement_rules/1
```

```
{
  "scope": [
    {"label": {"href": "/orgs/1/labels/1"}},
    {"label": {"href": "/orgs/1/labels/10"}},
    {"label_group": {"href": "/orgs/1/sec_policy/active/label_groups/8c6c50fc-4eac-
11ea-b687-ef0be7369e0c"}}
  ],
  "enforced_services": [
    {
      "href": "/orgs/1/sec_policy/draft/services/5"
    },
    {
      "port": 2456,
      "proto": 6
    }
  ],
}
```

Curl Command to Delete the Selective Enforcement Rule

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/3/sec_
policy/draft/selective_enforcement_rules/1
```

Virtual Server Filtering

Filtering of the discovered virtual servers and draft virtual servers endpoints makes it easier to manage large numbers of virtual servers.

The existing Public Experimental API endpoints for virtual servers have been changed to support the required filters and associated UI operations. You can now filter a discovered virtual server collection by:

- name
- SLB (API uses href as per conventions)

- VIP: IP, proto, port (any or all)
- virtual server href

Virtual Server Endpoints

New filters have been added for the following existing endpoints:

- GET /orgs/:xorg_id/discovered_virtual_servers
- GET /orgs/:xorg_id/sec_policy/:pversion/virtual_servers



NOTE:

These Interface endpoints are available only for API version V2.

New Filters for Virtual Servers

Filters for discovered_virtual_servers

Filter	URI Example and Notes
name	/discovered_virtual_servers?name=myvip Supports partial and incomplete matches.
slb	/discovered_virtual_servers?slb=/orgs/1/slbs/<uuid>
vip	/discovered_virtual_servers?vip=10.1 Supports suffix matches, e.g. 10.1 matches any IP address that starts with "10.1", "10.100", ... but not "110.x"
vip- proto	/discovered_virtual_servers?vip_proto=6
vip- port	/discovered_virtual_servers?vip_port=80
has_vir- tual_ server	/discovered_virtual_servers?has_virtual_server=true The virtual_server_mode and virtual_server_labels MUST be used with has_virtual_server=true, otherwise an error will be raised.
virtual_ server_ mode	/discovered_virtual_servers?virtual_server_mode=enforced Options for this filter are "unmanaged" or "enforced"
virtual_ server_ labels	/discovered_virtual_servers?virtual_server_labels=[[/orgs/1/labels/2, /orgs/1/labels/3], [/orgs/1/labels/4]] (JSON encoded array of arrays)
virtual_ server	/discovered_virtual_servers?virtual_server=/orgs/1/sec_poli- cy/draft/virtual_servers/<uuid>

Filters for virtual_servers

Filter	URI Example and Notes
name	/virtual_servers?name=myvip Supports partial and incomplete matches.
slb	/virtual_servers?slb=/orgs/1/slbs/<uuid>
vip	/virtual_servers?vip=10.1 Supports suffix matches, e.g. 10.1 matches any IP address that starts with "10.1", "10.100", ... but not "110.x."
vip-proto	/virtual_servers?vip_proto=6
vip_port	/virtual_servers?vip_port=80
mode	/virtual_servers?mode=enforced Options for this filter are "unmanaged" or "enforced."
labels	/virtual_servers?[[/orgs/1/labels/2, /orgs/1/labels/3], [/orgs/1/labels/4]] (JSON encoded array of arrays)
discovered_virtual_server	/virtual_servers?discovered_virtual_server=/orgs/1/discovered_virtual_servers/<uuid>

Schema Changes for Virtual Server Endpoints

Changes for discovered_virtual_servers

The following object has been added to the schema:

```
{
  [... existing fields ...]
  "virtual_server" : {
    "href" : "/orgs/1/sec_policy/draft/virtual_servers/fbae7cd2-04c3-4d7b-a628-2d69a9d64a71" ,
    "update_type" : "create" , # or "update" , "delete" , null
    "mode" : "enforced" , # or "unmanaged"
    "labels" [
      { "href" : "/orgs/1/labels/2" , "key" : "role" , "value" : "database"
      { "href" : "/orgs/1/label/12" , "key" : "env" , "value" : "production"
    ]
  }
}
```


Changes for virtual_servers

The "mode" and "vip_port" fields have been added to the "discovered_virtual_server" sub-object" to reflect the result of filtering.

```
{
  [... existing fields ...]
  "discovered_virtual_server" : {
    "dvs_identififier" : "5111ecf75c61544720d800cce97a624d" ,
    "href" : "/orgs/1/discovered_virtual_servers/c1cd1f00-7b48-4c43-a099-
f758ac1a9b40" ,
    "mode" : "snat" ,
    "name" : "Common/vip1" ,
    "vip_port" : {
      "port" : "80" ,
      "protocol" : 6 ,
      "vip" : "10.0.0.109"
    }
  }
}
```

Request and Response Examples

Curl Command for Discovered Virtual Servers

```
curl -i -u api_
1bbac8b7295e9b512:343461267jks009651245343461267jks00965124b27074fa181f1edb3bb4a3
https://2x2testvc27.ilabs.io:8443/api/v2/orgs/1/discovered_virtual_servers
```

Response Body

```
[{
  "href": "/orgs/1/discovered_virtual_servers/52044aea-14db-4510-a1c6-00231230034",
  "dvs_identififier": "96803bd07185cd093dd800231230034",
  "name": "Common/QL_VIP_1",
  "nfc": {
    "href": "/orgs/1/nfcs/0bcf6c3d-f588-44c7-a269-00231230034"
  },
  "slb": {
    "href": "/orgs/1/slbs/84a1cd93-142f-480d-b9f8-00231230034"
  }
}]
```

```
    },
    "vip_port": {
      "vip": "172.16.27.88",
      "protocol": 6,
      "port": "8080"
    },
    "local_ips": ["172.16.26.18", "172.16.27.18"],
    "mode": "snat",
    "snat_type": "snat_pool",
    "snat_pool_ips": ["172.16.26.27", "172.16.26.18", "172.16.27.18"],
    "service_checks": [{
      "protocol": 1
    }],
    "created_at": "2021-02-26T08:32:02.131Z",
    "updated_at": "2021-02-26T08:32:02.131Z",
    "created_by": {
      "href": "/orgs/1/nfcs/0bcf6c3d-f588-44c7-a269-00231230034"
    },
    "updated_by": {
      "href": "/orgs/1/nfcs/0bcf6c3d-f588-44c7-a269-00231230034"
    }, {
      "href": "/orgs/1/discovered_virtual_servers/073c40ec-7357-44f4-a66d-002312300349",
      "dvs_identifier": "b679034796cdde929a00231230034",
      "name": "Common/QL_VIP_2",
      "nfc": {
        "href": "/orgs/1/nfcs/0bcf6c3d-f588-44c7-a269-00231230034"
      },
      "slb": {
        "href": "/orgs/1/slbs/84a1cd93-142f-480d-b9f8-00231230034"
      },
      "vip_port": {
        "vip": "172.16.27.71",
        "protocol": 6,
        "port": "8080"
      },
      "local_ips": ["172.16.26.18", "172.16.27.18"],
```

```
"mode": "snat",
"snat_type": "snat_pool",
"snat_pool_ips": ["172.16.26.28", "172.16.26.18", "172.16.27.18"],
"service_checks": [{
  "protocol": 1
}],
"created_at": "2021-02-26T08:32:02.177Z",
"updated_at": "2021-02-26T08:32:02.177Z",
"created_by": {
  "href": "/orgs/1/nfcs/0bcf6c3d-f588-44c7-a269-00231230034"
},
"updated_by": {
  "href": "/orgs/1/nfcs/0bcf6c3d-f588-44c7-a269-00231230034"
}
}]
```

Curl Command for Virtual Servers

```
curl -i -u api_
1bcab8b7295e9b512:343461267jks00965124500jkjdmnwe00231230034dfd256124fa181f1edb3bb
4a3 https://2x2testvc27.ilabs.io:8443/api/v2/orgs/1/sec_policy/draft/virtual_
servers
```

Response Body

```
[{
  "href": "/orgs/1/sec_policy/draft/virtual_servers/5c7aeb96-56e2-4af8-8b4e-
00231230034",
  "created_at": "2021-02-26T08:38:15.298Z",
  "updated_at": "2021-02-26T08:39:21.676Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/1"
  },
  "updated_by": {
    "href": "/users/1"
  },
  "deleted_by": null,
  "update_type": null,
}
```

```
"name": "Common/QL_VIP_1",
"description": "",
"discovered_virtual_server": {
  "href": "/orgs/1/discovered_virtual_servers/52044aea-14db-4510-a1c6-00231230034"
},
"dvs_name": "Common/QL_VIP_1",
"dvs_identifer": "96803bd07185cd093dd800231230034",
"labels": [{
  "href": "/orgs/1/labels/1185",
  "key": "role",
  "value": "Database_VIP_1"
}, {
  "href": "/orgs/1/labels/1178",
  "key": "app",
  "value": "Application_1"
}, {
  "href": "/orgs/1/labels/1176",
  "key": "loc",
  "value": "test_place_1"
}, {
  "href": "/orgs/1/labels/1174",
  "key": "env",
  "value": "Production"
}],
"service": {
  "href": "/orgs/1/sec_policy/draft/services/1"
},
"providers": [{
  "label": {
    "href": "/orgs/1/labels/1183",
    "key": "role",
    "value": "Web"
  }
}, {
  "label": {
    "href": "/orgs/1/labels/1178",
    "key": "app",
    "value": "Application_1"
  }
}
```

```
    }  
  }, {  
    "label": {  
      "href": "/orgs/1/labels/1176",  
      "key": "loc",  
      "value": "test_place_1"  
    }  
  }, {  
    "label": {  
      "href": "/orgs/1/labels/1174",  
      "key": "env",  
      "value": "Production"  
    }  
  }  
}],  
"mode": "unmanaged"  
}]
```

RBAC for PCE Users

This chapter contains the following topics:

RBAC Overview	166
RBAC User Operations	168
Authorization Security Principals	174
RBAC Permissions	178
Organization-wide Default User Permissions	185
App Owner RBAC Role	188

As an Illumio administrator, use the Role-based Access Control (RBAC) API to assign privileges and responsibilities to users as follows:

- Establish the least required privileges to perform a job.
- Limit access to the smallest operation-set to perform a job.
- Separate users' duties, such as give the responsibility or delegate authority to a specific team.
- Allow access based on roles and scopes. Scopes in the Illumio Core specify the domain boundaries granted to a user.
- Manage user authentication and authorization.

RBAC Overview

The Role-based Access Control (RBAC) is a Public Experimental API that gets, creates, updates, or deletes permissions for users and groups. These users and groups are managed locally by the PCE or externally by a single sign-on (SSO) identity provider (IdP).

Before you begin using the RBAC feature with the REST API, learn about the Illumio Core permissions model and its terms and concepts.

RBAC Terms and Concepts

You should be familiar with the following RBAC terms before using this API:

User

A user is a PCE account that provides login or API access to the PCE. A user can be managed locally by the PCE or externally through an IdP.

Permission

A permission represents a combination of a user's account, an RBAC role, and an optional scope. You can grant multiple permissions to a user, depending on your requirements. A permission is a three tuple consisting of a role, a scope, and an authorization security principal:

- **Role:** User personas that are associated with a set of allowed operations, such as creating new labels or provisioning policy changes. Roles can be one of two general types: unscoped and scoped.
 - **Unscoped roles** (or roles with “global scopes”) do not have restrictions on the types of resources on which a user can operate. This means that the role is not affected by any label scopes.
 - **Scoped roles** use one or more unique application, environment, and location labels (each with a label HREF, key, and value), to restrict user or group permissions to only those objects that share the same labels. Specifically, scoped roles allow certain users to create rules and rulesets and provision them.
- **Scope:** A set of three labels (one of each type for Application, Environment, and Location) that restricts operations to those workloads sharing the same labels as the scope label set.
 - GET, POST, and PUT permissions methods for the Ruleset Manager (limited or full) or Ruleset Provisioner roles have a required scope parameter. When granting permissions, choose a scope that restricts which resources these users can use in a ruleset, or which resources they can provision.
 - A scope contains zero or more applications, environment, and location labels. Each label in the scope is identified by its HREF. A scope can also contain zero or more label groups.

- If the scope is an empty array ([]), it includes all applications, environments, and locations.
- If one of the label types is not specified, all instances of that type are permitted. For example, if application labels are omitted but environment and location labels are present, all applications are within the scope.
- **Authorization Security Principal:** The binding that connects a user account with its permissions (a role, and depending on the role, scopes).

**NOTE:**

If you are using an external identity provider to manage user access to the PCE, make sure that your identity provider is configured and those external users have been added to the PCE *before* you use this API to assign user permissions.

Grant Permissions Workflow

Granting user permissions with the REST API follows this general workflow:

1. **Create a local user (optional)**

This step creates a new local PCE user with no permissions and sends an e-mail invitation to the user's e-mail address. (If you use an external identity provider to manage user access to the PCE, skip this step.)

2. **Create an authorization security principal**

An authorization security principal serves as the binding between a user or a group and an RBAC role and optional scopes.

3. **Grant permissions by assigning a role and scopes to the authorization security principal**

Once a user account has been associated with an authorization security principal, you can assign an RBAC role to the account and add custom scopes if the user role requires them.

RBAC User Operations

This Public Stable API creates, updates, and re-invites local users, and converts a local user to an external user or converts an external user to a local user. This API is intended only for local users managed by the PCE, not users managed by an external identity provider (IdP).

RBAC Users API Methods

Functionality	HTTP	URI
Get a collection of users	GET	[api_version]/users
GET an individual user	GET	[user_href]
Create a local user and send an e-mail invitation	POST	[api_version]/users
Convert an external user to a local user	POST	[user_href]local_profile
Delete a local user and convert to an external user	DELETE	[user_href]local_profile
Re-invite a local user	PUT	[user_href]local_profile/reinvite
For authenticated users: change your password by sending a request to the agent service.	PUT	[user_href]local_profile/password

Get RBAC Users

These methods get a collection of users or an individual user in the organization.

By default, the maximum number of users returned from a GET collection is 500. If you want to get more than 500 users, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Local Users

```
GET [api_version]/users
```

URI to Get an Individual User

```
GET [user_href]
```

Query Parameters

Parameter	Description	Type
type	The type of user, either local or external	String

Curl Command Get Collection of Local Users

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/users?type=local -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

```
[
  {
    "href": "/users/99",
    "type": "local",
    "effective_groups": [],
    "id": 99,
    "username": "joe.user@example.com",
    "full_name": "Joe User",
    "time_zone": "America/Los_Angeles",
    "locked": false,
    "login_count": 1,
    "last_login_ip_address": "192.x.x.x",
    "last_login_on": "2016-03-11T08:19:17.587Z",
    "local_profile": { "pending_invitation": false },
    "created_at": "2016-03-08T20:58:05.882Z",
    "updated_at": "2016-03-11T08:19:17.588Z"
  }
  .....
  {
    "href": "/users/56",
    "type": "local",
    "effective_groups": [],
    "id": 56,
    "username": "jeff.user@example.com",
    "full_name": "Jeff User",
    "time_zone": "America/New_York",
    "locked": false,
    "login_count": 21,
    "last_login_ip_address": "192.x.x.x",
    "last_login_on": "2017-05-26T14:22:37.643Z",
    "local_profile": { "pending_invitation": true },
    "created_at": "2016-05-02T07:16:21.725Z",
    "updated_at": "2017-05-26T14:23:04.625Z"
  }
]
```

Pending Invitation

Users with "pending_invitation": "true" in the response have not yet accepted the invitation to log in and create an account.

```
{
  "href": "/users/56",
  "type": "local",
  "effective_groups": [],
  "id": 56,
  "username": "jeff.user@example.com",
  "full_name": "Jeff User",
  "time_zone": "America/New_York",
  "locked": false,
  "login_count": 21,
  "last_login_ip_address": "192.x.x.x",
  "last_login_on": "2017-05-26T14:22:37.643Z",
  "local_profile": { "pending_invitation": true },
  "created_at": "2016-05-02T07:16:21.725Z",
  "updated_at": "2017-05-26T14:23:04.625Z"
}
```

Create a Local User

This method creates local users who are managed by the PCE.

URI to Create a Local User

```
POST [api_version]/users
```

Request Properties

Property	Description	Required
username	Identify a local user by an e-mail address, which must meet these requirements: <ul style="list-style-type: none"> • Be unique • Use the format xxxx@yyyy.zzz • Be 255 characters or less 	Yes
type	Indicates that the user created is a local user managed by the PCE.	Yes
full_name	Full user name.	No

Property	Description	Required
timezone	User time zone IANA region name.	No

Request Body

```
{
  "username": "joe_user@mycompany.com",
  "display_name": "Joe User ",
  "type": "local"
}
```

Curl Command to Create a Local User

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/users -H "Content-Type: application/json" -u $KEY:$TOKEN -d '{"username": "joe_user@mycompany.com","display_name": "Joe User","type": "user"}'
```

Convert Local User to External User

This method converts a local user to an external user by *deleting* the local user account profile.

Use the user HREF, which is obtained from the response when a user logs into the PCE using the Login API or from the GET collection response.

For example: /users/14

URI to Convert a Local User to an External User

```
DELETE [user_href]/local_profile
```

Example

```
DELETE https://pce.my-company.com:8443/api/v2/users/14/local_profile
```

Curl Command Convert Local User to External User

```
curl -i -X >DELETE https://pce.my-company.com:8443/api/v2/users/14/local_profile -H "Accept: application/json" -u $KEY:$TOKEN
```

Convert External User to Local User

This method converts externally managed users to local users who are managed by the PCE.

URI to Convert an External User a Local User

```
POST [user_href]/local_profile
```

Example

```
POST https://pce.my-company.com:8443/api/v2/users/14/local_profile
```

Curl Command Convert External User to Local User

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/users/14/local_profile -H "Content-Type: application/json" -u $KEY:$TOKEN
```

Re-invite a Local User

If you have already created a local user, but that user has not logged in yet for the first time, you can use this method to resend the email invitation. Once they receive the invitation, they can log into the PCE and complete their PCE user account registration.

URI to Re-invite a Local User

```
PUT [user_href]/local_profile/reinvite
```

Example

```
PUT https://pce.my-company.com:8443/api/v2/users/14/local_profile/reinvite
```

Curl Command to Re-invite a Local User

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/users/14/local_profile/reinvite -H "Content-Type: application/json" -u $KEY:$TOKEN
```

Authorization Security Principals

This Public Experimental API gets, creates, updates, and deletes authorization security principals.

An authorization security principal connects a user account with its permissions, which consists of a role and optional scopes.

Authorization Security Principals API Methods

Functionality	HTTP	URI
Get a collection of authorization security principals in an organization	GET	[api_version][org_href]/auth_security_principals
Create an individual authorization security principal	POST	[api_version][org_href]/auth_security_principals
Get an individual authorization security principal	GET	[api_version][auth_security_principal_href]
Update an authorization security principal	PUT	[api_version][auth_security_principal_href]
Delete an authorization security principal	DELETE	[api_version][auth_security_principal_href]

Get Authorization Security Principals

This method gets an individual or a collection of authorization security principals in your organization.

By default, the maximum number returned from a GET collection of authorization security principals is 500. If you want to get more than 500, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Authorization Security Principals

```
GET [api_version][org_href]/auth_security_principals
```

URI to Get an Individual Authorization Security Principal

Use the `auth_security_principal_id` in a GET collection response (the last set of numbers in an HREF field).

```
GET [api_version][org_href]/auth_security_principals/{auth_security_principal_id}
```

Query Parameters

You can use the following optional query parameters to restrict the results of a GET collection call:

Parameter	Description	Type
name	Name of service to use to filter. This parameter supports partial matches.	String
type	One of two types of users, either user or group.	String

Curl Command to Get Authorization Security Principals

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/auth_security_principals -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

Each individual authorization security principal returned is identified by its HREF. You can use the HREF to GET, PUT, or DELETE an authorization security principal.

```
{
  "href": "/orgs/7/auth_security_principals/97cb9898-027b-474e-9807-19e04460dfb0",
  "name": "jimmyjo@illum.io",
  "display_name": "Jimmy Joe Meeker",
  "type": "user"
},
.....
{
  "href": "/orgs/7/auth_security_principals/db7a2657-dcb8-4237-a6e7-7269cdbaea5d",
  "name": "foxy.brown@illumio.com",
  "display_name": "Foxy Brown",
  "type": "user"
}
]
```

Curl Command to Get an Authorization Security Principal

```
curl -i -X GET -H "Accept: application/json -u $KEY:'TOKEN' https://pce.my-company.com:8443/api/v2/orgs/2/auth_security_principals/db7a2657-dcb8-4237-a6e7-7269cdbaea5d
```

Create an Authorization Security Principal

This method creates an individual authorization security principal.

URI to Create an Authorization Security Principal

```
POST [api_version][org_href]/auth_security_principals
```

Request Body

Property	Description	Type	Required
type	One of two types, either user or group.	String	Yes
name	Name of the authorization security principal. <ul style="list-style-type: none"> If the user is local (managed by the PCE), the name must be an e-mail address of the local user. If the user or group are managed by an external IdP, use the name that identifies the external user or group in the external system. 	String	Yes
display_name	An optional display name for the authorization security principal.	String	No

Request Body - Local User Authorization Security Principal

```
{
  "type": "user",
  "name": "joe_user@illumio.com",
  "display_name": "Joe User"
}
```


Response Body - Local User Authorization Security Principal

```
{
  "href": "/orgs/7/auth_security_principals/e8c232d2-e4bf-4ba5-bd77-ccfc3a8ad999",
  "name": "joe_user@illumio.com",
  "display_name": "Joe User",
  "type": "user"
}
```

Request Body - External Group User Authorization Security Principal

```
{
  "type": "group",
  "name": "jCQN=Bank-Admin,OU=EU,DC=Acme,DC=com",
  "display_name": "Provisioners for Bank Accounts"
}
```

Response Body - External Group Authorization Security Principal

```
{
  "href": "/orgs/7/auth_security_principals/e8c232d2-e4bf-4ba5-bd77-ccfc3a8ad777",
  "name": "jCQN=Bank-Admin,OU=EU,DC=Acme,DC=com",
  "display_name": "Acme Bank Admins",
  "type": "group"
}
```

Curl Command Create an Authorization Security Principal

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/auth_security_principals -u $KEY:$TOKEN -H "Content-Type:application/json" -d '{"type": "user", "name": "joe_user@illumio.com", "display_name": "Joe User"}'
```

Update an Authorization Security Principal

In order to update an individual authorization security principal, use its HREF, which is obtained from the response from a GET collection.

URI to Update an Individual Authorization Security Principal

```
PUT [api_version][auth_security_principal_href]
```

Request Body

```
{  
  "type": "user",  
  "name": "joe_user2@illumio.com",  
  "display_name": "Joe User"  
}
```

Curl Command Create an Authorization Security Principle

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/sec_  
policy/draft/services/79 -H "Content-Type:application/json" -u $KEY:$TOKEN -d '  
{"type": "user", "name": "joe_user2@illumio.com", "display_name": "Joe User"}'
```

Delete an Authorization Security Principal

To delete an authorization security principal, use its HREF, which is returned in the response from a GET collection.

URI to Delete an Individual Authorization Security Principal

```
DELETE [api_version][auth_security_principal_href]
```

Curl Command Delete the Authorization Security Principal

```
curl -i -X DELETE -H "Accept: application/json" -u $KEY:$TOKEN https://pce.my-  
company.com:8443/api/v2/orgs/2/auth_security_principals/e8c232d2-e4bf-4ba5-bd77-  
ccfc3a8ad777
```

RBAC Permissions

This Public Experimental API grants permissions to PCE users and groups. It also returns a collection of permissions in the organization, gets individual user permissions, and updates and deletes permissions.

RBAC Permissions API Methods

Functionality	HTTP	URI
Get a list of all RBAC permissions for the organization	GET	[api_version][org_href]/permissions
Get an individual permission	GET	[api_version][permission_href]
Grant a permission	POST	[api_version][permission_href]
Update a permission	PUT	[api_version][permission_href]
Delete a permission	DELETE	[api_version][permission_href]

Parameters for Roles

Unscoped Roles

API Role Name	UI Role Name	Granted Access
owner	Global Organization Owner	Perform all actions: Add, edit, or delete any resource, security settings, or user accounts.
admin	Global Administrator	Perform all actions except cannot change security settings and cannot perform user management tasks.
read_only	Global read-only	View any resource or security settings. Cannot perform any operations.
global_object_provisioner	Global Policy Object Provisioner	Provision rules containing IP lists, services, and label groups, and manage security settings. Cannot provision rulesets, virtual services, or virtual servers, or add, modify, or delete policy items.

Scoped Roles

API Role Name	UI Role Name	Granted Access
ruleset_manager	Full Ruleset Manager	Add, edit, and delete all rulesets within a specified scope. Add, edit, and delete rules when the provider matches a specified scope. The rule consumer can match any scope.
limited_ruleset_manager	Limited Ruleset Manager	Add, edit, and delete all rulesets within a specified scope. Add, edit, and delete rules when the provider and consumer match the specified scope. Ruleset managers with limited privileges cannot manage rules that use IP lists,

API Role Name	UI Role Name	Granted Access
		user groups, label groups, iptables rules as consumers, or rules that allow internet connectivity.
ruleset_provisioner	Ruleset Provisioner	Provision rulesets within a specified scope. Cannot provision virtual servers, virtual services, SecureConnect gateways, security settings, IP lists, services, or label groups.

Scopes for Ruleset Manager and Ruleset Provisioner

If you are granting a user or group the Ruleset Manager or the Ruleset Provisioner role, you can also associate a scope to the role so you can control which rulesets they can add and provision.

There is a default read-only user permission that is organization-wide and inherited by all users in the organization. This global permission allows users who have no permissions explicitly granted to them to access the PCE.



NOTE:

For information, see [Organization-wide Default User Permissions](#).

Role HREF Syntax

An RBAC role is identified in the REST API by its HREF, the exact syntax of which is based on the PCE organization HREF [org_href].

```
[org_href]/roles/[role_name]
```

For example, if you wanted to grant a user permission with the Global Object Provisioner role, and your PCE organization HREF is /org/6, the role HREF would look like:

```
/orgs/6/roles/global_object_provisioner
```

Get RBAC Permissions

These methods get an individual user permission or a collection of permissions in the organization.

By default, the maximum number of permissions returned on a GET collection is 500. If you want to get more than 500, use an [Asynchronous GET Collection](#).

URI to Get All Permissions in Your Organization

```
GET [api_version][org_href]/permissions
```

URI to Get an Individual Permission

```
GET [api_version][permissions_href]
```

Query Parameters for Permissions

Parameter	Description	Type	Required
org_id	Organization	Integer	Yes
permission_id	UUID of the permission. Used to get, update, and delete an individual permission	String	Yes
auth_security_principal	The authorization security principal associated with the permission. It is not needed to get an individual permission or to delete a permission.	String	POST:Yes PUT:No
role	<p>The RBAC role associated with the permissions.</p> <p>(For additional information about these roles and their associated capabilities, see the <i>PCE Administration Guide</i>.)</p> <p>Unscoped roles:</p> <ul style="list-style-type: none"> owner admin read_only global_object_provisioner <p>Scoped roles:</p> <ul style="list-style-type: none"> ruleset_manager limited_ruleset_manager ruleset_provisioner 	String	Yes
scope	See Parameters for Roles .	String	Yes

Curl Command Get Permissions with a Specific Role

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/permissions?role=ruleset_provisioner -H "Accept: application/json" -u $KEY:$TOKEN
```

Grant RBAC Permissions

When an RBAC permission is granted to a user in the PCE, the user account (identified by its authorization security principal) is associated with a role. Depending on the role, scopes can be applied that restrict the permission to operating on specified labeled resources.

URI to Create a New Permission

```
POST [api_version][org_href]/permissions
```

Permission Parameters

Property	Description	Required
role	<p>The HREF of the role to assign to this user or group.</p> <p>An RBAC role is identified in the REST API by its HREF, the exact syntax of which is different for every user and is based on the PCE organization HREF [org_href].</p> <p>For example: [org_href]/roles/[role_name]</p> <p>For example, to grant a user permission with the Global Object Provisioner role, with a PCE organization HREF of /org/6, the role HREF would be:</p> <p>/orgs/6/roles/global_object_provisioner</p> <p>For a complete list of role names and their descriptions, see the <i>PCE Administration Guide</i>.</p>	Yes
auth_security_principal	<p>The HREF of the authorization security principal (auth_security_principal) associated with the user or group being granted a permission.</p> <p>The HREF of an authorization security principal is returned when you create a new one, or you can GET a collection of</p>	Yes

Property	Description	Required
	authorization security principals in your PCE.	
scope	See Parameters for Roles	

Request Body - Grant a Scoped Permission

The permission for this scoped role consists of the following elements:

- A scope for the role (application, environment, and location labels)
- The role
- An authorization security principal associated with a user account

```
{
  "scope": [
    {"href": "/orgs/7/labels/91", "key": "app", "value": "db"},
    {"href": "/orgs/7/labels/92", "key": "loc", "value": "nyc"},
    {"href": "/orgs/7/labels/100", "key": "env", "value": "prod"}
  ],
  "role": { "href": "/orgs/7/roles/ruleset_provisioner" },
  "auth_security_principal": {"href": "/orgs/7/auth_security_principals/xxxxxxx-
e4bf-4ba5-bd77-ccfc3a8ad999"}
}
```

Response - Scoped Permission

The response shows the new permission (at the top) that has been created identified by its HREF:

```
{
  "href": "/orgs/7/permissions/51d9207c-354b-45de-9bf5-d1b613ac3719",
  "role": { "href": "/orgs/7/roles/ruleset_provisioner" },
  "scope": [
    {"href": "/orgs/7/labels/91", "key": "app", "value": "db"},
    {"href": "/orgs/7/labels/92", "key": "loc", "value": "nyc"},
    {"href": "/orgs/7/labels/100", "key": "env", "value": "prod"}
  ],
  "auth_security_principal": {"href": "/orgs/7/auth_security_principals/xxxxxxx-
e4bf-4ba5-bd77-ccfc3a8ad999"}
}
```

Request Body - Unscoped Permission

In this request for an unscoped permission, the required scope property is defined as an empty JSON array ([]).

```
{
  "scope": [],
  "role": { "href": "/orgs/7/roles/owner" },
  "auth_security_principal":{"href":"/orgs/7/auth_security_principals/xxxxxxx-
e4bf-4ba5-bd77-ccfc3a8ad999"}
}
```

Response - Unscoped Permission

```
{
  "href": "/orgs/7/permissions/51d9207c-354b-45de-9bf5-d1b613ac3719",
  "role": { "href": "/orgs/7/roles/owner" },
  "scope": [],
  "auth_security_principal":{"href":"/orgs/7/auth_security_principals/xxxxxxx-
e4bf-4ba5-bd77-ccfc3a8ad999"}
}
```

Update an RBAC Permission

This method updates a permission, for example changing the permission role, authorization security principal, user, or group.

URI to Update a Permission

```
PUT [api_version][permissions_href]
```

Curl Command to Update the Role Permission

```
curl -i -X PUT https://pce.mycompany.com:8443/api/v2/orgs/7/permissions/xxxxxxx-
354b-45de-9bf5-d1b613ac3719 -H "Content-Type: application/json" -u $KEY:$TOKEN -d
'{"scope": [{"href": "/orgs/7/labels/91", "key": "app", "value": "db"}, {"href":
"/orgs/7/labels/92", "key": "loc", "value": "nyc"}, {"href": "/orgs/7/labels/100",
"key": "env", "value": "prod"}], "role": {"href": "/orgs/7/roles/global_object_
provisioner"}, "auth_security_principal":{"href":"/orgs/7/auth_security_
principals/xxxxxxx-e4bf-4ba5-bd77-ccfc3a8ad999}}'
```


Delete an RBAC Permission

Curl Command to Delete a Permission

```
curl -i -X DELETE
https://pce.mycompany.com:8443/api/v2/orgs/7/.permissions/xxxxxxx-354b-45de-9bf5-
d1b613ac3719 -H "Accept: application/json-u $KEY:$TOKEN"
```

Organization-wide Default User Permissions

This Public Experimental API supplies an organization-wide default user permission and allows users to log into the PCE and view resources. These resources don't have to be explicitly assigned to any RBAC roles or scopes.

About Default User Permissions

If you use an external identity provider for user management, you might want to block some of those users from the PCE without removing them from your identity provider. *Deleting* the organization-wide read-only permission allows you to achieve this.

When the read-only user permission is disabled for your organization, users who are not explicitly assigned this permission cannot log into the PCE and access Illumio resources. If users without permissions attempt to log into the PCE, their external identity provider authenticates them but the PCE immediately logs them out.

To disable organization-wide read-only permissions:

1. Get a collection of all authorization security principals in your organization, and search the response for the one named `nu11`. Once you find this authorization security principal, make a note of its full HREF.
2. Get the HREF of the permissions object associated with the `nu11` authorization security principal. Keep a record of the JSON object for this permission in the event you want to re-enable the permission at a later date.
3. Delete the permission associated with the `nu11` authorization security principal.

Get a Collection of Authorization Security Principals

The first step in disabling the organization-wide read-only permission is to get a collection of all authorization security principals in your organization.

Curl Command Get Auth Security Principals Collection

```
curl -i -X GET https://pce.mycompany.com:8443/api/v2/orgs/7/auth_security_
principals -H "Accept: application/json" -u $KEY:$TOKEN
```

Example Response Body

The null authorization security principal in the following example is highlighted in blue:

```
[
.....
  {
    "href": "/orgs/7/auth_security_principals/a23ea011-4191-49e6-a22a-d3dba4fb8058",
    "name": null,
    "display_name": null,
    "type": "group"
  },
.....
]
```

Get Permission for the Null Auth Security Principal

To get the permission object associated with the null authorization security principal, call the GET Permissions API with the query parameter value set to the HREF for the null authorization security principal similar to curl command:

```
curl -i -X GET -H "Accept: application/json" -u $KEY:$TOKEN
https://pce.mycompany.com:8443/api/v2/orgs/7/permissions?auth_security_
principal=/orgs/7/auth_security_principals/a23ea011-4191-49e6-a22a-d3dba4fb8058
```

Response

The response returns the HREF of the permission associated with the organization-wide read-only permission.

```
{
  "href": "/orgs/7/permissions/14c92849-e88e-4930-8804-3245565619e5",
  "role": {
    "href": "/orgs/7/roles/read_only"
```

```
  },
  "scope": [],
  "auth_security_principal": {
    "href": "/orgs/7/auth_security_principals/a23ea011-4191-49e6-a22a-
d3dba4fb8058"
  }
}
```

Delete Null Authorization Security Principal Permission

Keep a record of the permission object returned in case you want to re-enable the permission in the future.

Delete the read-only permission HREF to disable it.

Curl Command to Delete Null Authorization Security Principal Permission

```
curl -i -X DELETE -H "Accept: application/json" -u $KEY:$TOKEN
https://pce.mycompany.com:8443/api/v2/orgs/7/permissions?auth_security_
principal=/orgs/7/auth_security_principals//orgs/7/permissions/14c92849-e88e-4930-
8804-3245565619e5
```

Response

An HTTP 200 response is returned on the successful deletion of the organization-wide read-only permission.

Re-Enable the Organization Read-Only Permission

If the organization-wide read-only permission was disabled, you can re-enable it by recreating the permission object. This object must be constructed exactly as the object that was returned to you when you got the permission. The request body below illustrates the JSON structure of this permission object.

URI to Enable the Organization-Wide Read-Only Permission

```
POST [api_version][permission_href]
```

Request Body

```
{
  "role": {
```

```
    "href": "/orgs/7/roles/read_only"
  },
  "auth_security_principal": {
    "href": "/orgs/7/auth_security_principals/a23ea011-4191-49e6-a22a-
d3dba4fb8058"
  },
  "scope": []
}
```

Curl Command to Enable Organization Read-Only Permission

```
curl -i -X POST https://pce.mycompany.com:8443/api/v2/orgs/7/permissions -H
"Content-Type: application/json" -u $KEY:$TOKEN -d '{"role": {"href":
"/orgs/7/roles/read_only"}, "auth_security_principal":{"href":"/orgs/auth_
security_principals/a23ea011-4191-49e6-a22a-d3dba4fb8058"}, "scope": []}'
```

Response

An HTTP 201 response is returned on successfully recreating the organization-wide read-only permission.

App Owner RBAC Role

The App Owner RBAC (Role-based Access Control) role hides information in the PCE that is not relevant to the user with that role. At the same time, the App Owners can write effective rules to secure their apps, as well as restrict visibility within the PCE to the permitted scopes for users.

RBAC was previously restricting only the write permission for users while the read permission was unrestricted, and every user had visibility into PCE. The App Owner RBAC role also restricts the read permission to correspond to the user roles. It accelerates enterprise-wide expansion so that the customers who acquired Illumio for a single application can expand faster

Introduction of the App Owner role solves these problems:

- It accelerates micro-segmentation deployment by allowing for scaling after an organization implements micro-segmentation with a smaller set of applications.
- It assures compliance with good security practices so that users cannot view the sensitive information they are not allowed to see.

- It eliminates the complexity of building a custom portal. The App Owners can use Illumio REST APIs instead of the custom UIs created by customers.

App Owners are responsible for managing vulnerabilities in the applications they own and for which the PCE owners can assign scoped roles.

App Owner Roles

Roles of Ruleset Managers, Ruleset Provisioners, and Workload Managers are assigned to users and user groups. They can be expanded with additional to provide the users with additional read/write permissions. All permissions are additive.

Ruleset Manager with Scoped Reads

This RBAC role has the write permission that allows its owner to make changes to the policy. Users with this role can see in the PCE only the content related to their location instead of having full read-only access to the entire PCE content as before.

The role now also supports scoped reads.

Ruleset Provisioner with Scoped Reads

This RBAC role can provision policy changes to workloads. Users with this role can see in the PCE only the content related to their location instead of having full read-only access to the entire PCE content as before.

The role now also supports scoped reads.

Ruleset Viewer

This RBAC role has access to the PCE to manage one or multiple applications. Users with this role can get a view of their application and its dependencies, but they cannot see information about other applications.

Workload Manager with Scoped Reads

This RBAC role provides a control for managing workloads. Users with this role can see in the PCE only the content related to their scope instead of having full read-only access to the entire PCE content as before.

The role now also supports scoped reads.

Rulesets and Rules

This chapter contains the following topics:

Rulesets	191
Rules	200
Custom iptables Rules	217
Machine Authentication	225

Illumio security policy includes three rule types: intra-scope rules, extra-scope rules, and custom iptables rules. The scope of a ruleset determines which workloads receive the ruleset's rules:

- Intra-scope rules allow communication between providers and consumers within a specific scope.
- Extra-scope rules permit communication between applications. You can write rules so that consumers within or outside a specified scope can access the providers within a scope. For extra-scope rules, the labels used in the scope must match the labels used by the provider.
- Custom iptables rules are needed for your applications as part of the rules managed by the PCE. These rules help preserve any configured iptables from native Linux host configurations by allowing you to include them with the rules for your policy.

You can combine multiple types of rules in a single ruleset.

Rulesets

This Public Stable API gets, creates, updates, and deletes rulesets. Rulesets contain rules and scopes, which define where the rules apply.

Ruleset API Methods

Functionality	HTTP	URI
Get a collection of rule-sets	GET	[api_version][org_href]/sec_policy/pversion/rule_sets
Get a specified ruleset	GET	[api_version][ruleset_href]
Create a ruleset	POST	[api_version][org_href]/sec_policy/draft/rule_sets
Update a specified rule-set	PUT	[api_version][ruleset_href]
Delete a specified rule-set	DELETE	[api_version][ruleset_href]

Active vs Draft

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, firewall settings, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID of the security policy.

Ruleset Components

Rulesets are the core of the Illumio Core policy model, and consist of the following elements:

- **Scopes:** Sets of labels (application, environment, and location) that define the boundaries of the rules in a ruleset. If the workloads specified in the rules share the same labels in a ruleset scope, then those workloads and their

communications are governed by the rules of the ruleset.

A scope can contain zero or more application, environment, and location labels. A scope can also contain one or more label groups.

If the scope is an empty array ([]), then the scope includes all applications, environments, and locations.

If one of the label types is not specified, then all instances of that type are permitted. For example, if application labels are omitted but environment and location labels are present, then all applications are within the scope.

A label type cannot be used in a rule unless the scope for the label type is “All.” For example, to use a location label, the scope would have to be an empty array ([]), or if there is an application label and an environment label in the scope, the location label cannot be defined in the scope.

A ruleset is not limited to a single scope. A rule can contain multiple scopes depending on the needs of the security policy.



IMPORTANT:

Role labels are not used in scopes, but can be used in rules. Never use a role label in a scope.

- **Rules:** A security rule consisting one or more providers (provides a service over a port and protocol), one or more consumers (consumes the service offered by the provider), and one or more services. A provider or consumer can be an individual workload, a role label that represents multiple workloads, IP lists, and so on.

Example Ruleset Scope

Each label in a scope is identified by its HREF. For example, this is the JSON representation of a single ruleset scope with three labels.



NOTE:

Each label must have a different key (role, app, loc, or env). Duplicate label keys are allowed in a scope only if they are in a label group.

```
{  
  "scopes": [  
    [  

```



```

    {"label": {"href": "/orgs/7/labels/105"}},
    {"label": {"href": "/orgs/7/labels/88"}},
    {"label": {"href": "/orgs/7/labels/98"}}
  ]
]
}
```

Ruleset Rules

Ruleset rules define the allowed communication between workloads, or between workloads and IP lists.

For information, see [Rules](#).

Get Rulesets

This method gets all of the rulesets in your organization. This method gets those rulesets that are in the “draft” policy state, which means the current state of rulesets that have not been provisioned.

By default, the maximum number returned on a GET collection of rulesets is 500.



NOTE:

To return more than 500 rulesets, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Rulesets

pversion: Contains provisionable objects, which exist in either a draft (not provisioned) or active (provisioned) state. .

```
GET [api_version][org_href]/sec_policy/:pversion/rule_sets
```

URI to Get an Individual Ruleset

```
GET [api_version][ruleset_href]
```

Query Parameters

You can use the following query parameters to restrict the results of the query to get a collection of rulesets.

Parameter	Description	Type	Required
enabled	Enabled flag	Boolean	Yes

Parameter	Description	Type	Required
name	Name of the rulesets to filter. This parameter supports partial matches.	String	Yes
scopes	Rule set scopes <ul style="list-style-type: none"> label: label URI label_group: label group URI 	Array String	Yes
rules	Array of rules in this rule set Required properties: enabled: Enabled flag providers: <code>\$ref": sec_policy_rule_sets_sec_rules_providers.schema</code> consumers: <code>\$ref": sec_policy_rule_sets_sec_rules_consumers.schema</code> ingress_services: <code>\$ref": sec_rule_ingress_services.schema</code> resolve_labels_as: <code>\$ref": sec_rule_resolve_labels_as.schema</code>	Array Boolean	Yes
created_at	Timestamp when this rule set was first created	String	Yes
updated_at	Timestamp when this rule set was last updated	String	Yes
deleted_at	Timestamp when this rule set was deleted	String	Yes
created_by	User who originally created this rule set	String	No
updated_by	User who last updated this rule set	String	No
deleted_by	User who deleted this rule set	String	No
update_type	Type of update	String	No
external_data_reference	A unique identifier within the external data source. For example, if ruleset information is stored in an external database.	String	No
external_data_set	The data source from which the resource originates. For example, if ruleset information is stored in an external database.	String	No
ip_tables_rules	Array of iptables rules in this rule set.	Array	No

Curl Command to Get Rulesets



NOTE:

To get all active rulesets in the current security policy, use `active` instead of `draft`.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/sec_
policy/draft/rule_sets -H "Accept: application/json" -u $KEY:$TOKEN
```

Curl Command to Get a Ruleset

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/sec_
policy/draft/rule_sets/90 -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

The following response shows an example of rulesets that are returned from a get collection of rulesets. Each ruleset is defined as an HREF (the first property of the ruleset in the response). Use the HREF of a ruleset to get, update, or delete an individual ruleset.

```
[
  {
    "href": "/orgs/1/sec_policy/draft/rule_sets/90",
    "created_at": "2019-05-11T21:55:22.930Z",
    "updated_at": "2019-06-14T18:08:59.134Z",
    "deleted_at": null,
    "created_by": {"href": "/users/1"},
    "updated_by": {"href": "/users/1"},
    "deleted_by": null,
    "name": "Rset",
    "description": "",
    "external_data_set": "",
    "external_data_reference": "",
    "enabled": true,
    "scopes": [
      [
        {"label": {"href": "/orgs/1/labels/11"}},
        {"label": {"href": "/orgs/1/labels/9"}}
      ]
    ]
  }
]
```

```
    {"label": {"href": "/orgs/1/labels/18"}}
  ]
],
"rules": [
  {
    "href": "/orgs/1/sec_policy/draft/rule_sets/90/sec_rules/111",
    "created_at": "2019-06-13T21:19:58.078Z",
    "updated_at": "2019-06-13T21:19:58.078Z",
    "deleted_at": null,
    "created_by": {"href": "/users/1"},
    "updated_by": {"href": "/users/1"},
    "deleted_by": null,
    "update_type": null,
    "description": null,
    "enabled": true,
    "providers": [
      {"label": {"href": "/orgs/1/labels/17"}}
    ],
    "consumers": [
      {"label": {"href": "/orgs/1/labels/1"}}
    ],
    "consuming_security_principals": [],
    "sec_connect": false,
    "stateless": false,
    "machine_auth": false,
    "unscoped_consumers": false,
    "ingress_services": [],
    "resolve_labels_as": {
      "providers": ["virtual_services"],
      "consumers": ["workloads"]
    },
  }
],
"ip_tables_rules": [],
"caps": ["write"]
}
```

Create a Ruleset

This method creates an individual ruleset. The PCE web console supports up to 500 rules per ruleset.



NOTE:

To write more than 500 rules for a particular ruleset, create additional rulesets, or use the Illumio Core REST API (rulesets with more than 500 rules are not fully displayed in the PCE web console).

URI to Create a Ruleset

```
POST [api_version][ruleset_href]
```

Required Properties for POST

Property	Description	Type	Required
name	Name of the new ruleset, which must be unique.	String	Yes
scopes	<p>Indicate the boundaries of the rules in the ruleset by specifying a set of one or more unique label scopes. Each scope must include the label HREF (for example, /orgs/1/labels/24).</p> <ul style="list-style-type: none"> • app: Application label or label group • env: Environment label or label group • loc: Location label or label group <p>You can also use a label group for a ruleset scope, and each label of the specific type (app, env, or loc) and all labels in each label group are used for the ruleset scope.</p> <p>To create a ruleset with scope=All/All/All, use an empty arrays for pb and ub scopes.</p>	Array	Yes

Request Body

This example illustrates the request body for creating a new ruleset with three scopes and two [Intra-scope rules: Allow communication between providers and consumers within a specific scope.](#) .

```

{
  {
    "name": "Demo RS",
    "enabled": true,
    /* Two ruleset scopes, each with an application, an environment,
       and a location label. To have more than one type of label in a scope,
       use a label group that contains only labels of that type. */
    "scopes": [
      [
        {"label": {"href": "/orgs/1/labels/24"}},
        {"label": {"href": "/orgs/1/labels/27"}},
        {"label": {"href": "/orgs/1/labels/21"}}
      ],
      [
        {"label": {"href": "/orgs/1/labels/15"}},
        {"label": {"href": "/orgs/1/labels/16"}},
        {"label": {"href": "/orgs/1/labels/17"}}
      ]
    ],
    "rules": [
      /* Label to label, intra-scope. */
      {
        "enabled": true,
        "providers": [{"label": {"href": "/orgs/1/labels/2"}},
        "consumers": [{"label": {"href": "/orgs/1/labels/1"}},
        "consuming_security_principals": [],
        "ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/20"}],
        "resolve_labels_as": {
          "providers": ["workloads"],
          "consumers": ["workloads"]
        },
        "sec_connect": false,
        "unscoped_consumers": false
      },
      /* This illustrates a provider with multiple labels and an IP list as a
         consumer.
         Note that both provider labels must be role labels in this context, because
         the ruleset scope already has application, environment, and location labels. */
    ]
  }
}
    
```


Delete a Ruleset

To delete an individual ruleset, you need the HREF of the ruleset you want to delete, which can be obtained when you get a collection of rulesets.

URI to Delete an Individual Ruleset

```
DELETE [api_version][ruleset_href]
```

Curl Command to Delete Ruleset

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/draft/rule_sets/179 -u $KEY:$TOKEN
```

Rules

This Public Stable API creates, updates, and deletes individual rules in rulesets . It also gets a collection of rules from a ruleset.

Rules API Methods

Functionality	HTTP	URI
Get a collection of rules from a rule-set	GET	[api_version][rule_set_href]/sec_rules
Get an individual rule from a ruleset	GET	[api_version][sec_rule_href]
Create rules	POST	[api_version][rule_set_href]/sec_rules
Update an individual rule	PUT	[api_version][sec_rule_href]
Delete an individual rule	DELETE	[api_version][sec_rule_href]

Active vs Draft

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, firewall settings, and virtual servers. For these objects, the URL of the API call must include the element called :pversion, which can be set to either draft or active.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be draft, active, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be draft (you cannot operate on active items) or the ID of the security policy.

Rule Types

There are three types of rules:

- **Intra-scope rules:** Allow communication between providers and consumers within a specific scope.
- **Extra-scope rules:** Rules that go beyond the scope of the ruleset to which they belong. In this rule type, the workloads, labels or IP list in the consumers part of the rule are not constricted by the scope of the ruleset. This type of rule is used when you want specific rules that allow providers to offer a service to other workloads or groups that are not within the boundaries of the ruleset scope.
- **Custom iptables rules:** Used to configure custom iptables rules on Linux workloads; for example, to preserve existing native Linux host iptables rules by including them in a ruleset.



NOTE:

The PCE web console can only display up to 500 rules per ruleset. To write more than 500 rules for a particular scope, consider splitting the rules across multiple rulesets, otherwise users won't be able to view them all in the PCE web console.

Rule Type JSON Specification

To define a rule as either intra-scope or extra-scope, specify if the rule is “scoped” or “not scoped” by defining the `'unscoped_consumers'` property:

- When a rule has `unscoped_consumers: false`, this defines an intra-scope rule, which means both its providers and consumers are bound by the ruleset scope.
- When a rule has `unscoped_consumers: true`, this defines an extra-scope rule, which means its providers are bound by the ruleset scope, but the consumers are *not* bound by the ruleset scope.

Intra-Scope Rule Example

This rule illustrates an intra-scope rule because it has its `unscoped_consumers` property set to `false`:

```
{
  "rules": [
    {
      "enabled": true,
      "providers": [ {"label": {"href": "/orgs/1/labels/2"} } ],
      "consumers": [ {"label": {"href": "/orgs/1/labels/1"} } ],
      "consuming_security_principals": [],
      "ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/20"}],
      "resolve_labels_as": {
        "providers": ["workloads"],
        "consumers": ["workloads"]
      },
      "sec_connect": false,
      "unscoped_consumers": false
    }
  ]
}
```

Providers and Consumers

The Illumio Core allowlist policy model uses rules to define the allowed communications between two or more workloads, or between workloads and other entities, such as IP lists, virtual servers, and the internet.

The fundamental structure of a rule (except custom iptables rules) consists of a provider, a service that the provider makes available over a network port and protocol, and a consumer of that service.

A simple rule that allows two workloads to communicate with each other might look like this:

Providers	Service	Consumers
Database	HTTPS	Web

This example shows the rule providers, consumers, and services:

```
{
  "enabled": true,
```

```
"providers": [ {"label": {"href": "/orgs/1/labels/10"} } ],
"consumers": [ {"label": {"href": "/orgs/1/labels/67"} } ],
"consuming_security_principals": [],
"ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/32"}],
"resolve_labels_as": {
  "providers": ["workloads"],
  "consumers": ["workloads"]
}
"sec_connect": false,
"unscoped_consumers": false
}
```

Stateless Rules

A rule can be configured to have stateless packet filtering (`"stateless": true`). This means that the VEN instructs the host firewall to *not* maintain persistent connections for all sessions. This type of rule is typically used for datacenter “core services” such as DNS and NTP.

A stateless rule can have these consumer types:

- Any IP list plus all workloads
- A label (one of a specific type)
- An individual item (such as an individual workload)

An attempt to add more consumers, or one not supported, will return an error.

A PCE can only have a maximum of 100 stateless rules. If an implementation requires more than 100 stateless rules, contact your Illumio Professional Services Representative for more information.



NOTE:

This property has an API exposure level of Public Experimental, which means it is not intended for production use and might change in future releases.

For more information, see [API Classification and Version](#).

Get Rules

This API gets a collection of rules or gets an individual rule from a ruleset.

Before you can get rules from a ruleset with this API, you need to obtain the ruleset HREF, which is returned when you [Get a Collection of Rulesets](#).

Example Ruleset HREF

```
/orgs/2/sec_policy/draft/rule_sets
```


URI to Get a Collection of Rules from a Ruleset

```
GET [api_version][rule_set_href]/sec_rules
```

URI to Get an Individual Security Rule from a Ruleset

```
GET [api_version][sec_rule_href]
```

Query Parameters to Get a Collection of Security Rules from a Ruleset

Parameter	Description	Type
pversion	Security policy version -- draft(not provisioned) or active (provisioned)	String
	 NOTE: This is a path parameter.	
description	Description of rulesets to return. Supports partial matches.	String
external_data_reference	A unique identifier within the external data source. For example, if this rule information is stored in an external database.	String
external_data_set	The data source from which the resource originates. For example, if this rule information is stored in an external database.	String
labels	List of lists of label URIs encoded as a JSON string.	String
max_results	Maximum number of rule sets to return.	Integer
name	Name of rulesets to return.	String

Query Parameters to Get an Individual Security Rule from a Ruleset

Parameter	Description	Type
:pversion	Security policy version -- draft (not provisioned) or active (provisioned).	String

Parameter	Description	Type
	This is a path parameter.	
rule_set_id	The ruleset ID. This is a path parameter.	Integer
representation	Representation details for this resource on the response object.	String

Curl Command to Get Rules from Ruleset

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/draft/rule_sets/97/sec_rules -H "Accept: application/json" -u $KEY:$TOKEN
```

Response Body

In the response, each returned rule is identified by its HREF, such as: `"/orgs/1/sec_policy/draft/rule_sets/152/sec_rules/124"`.

For example:

```
{
  "href": "/orgs/1/sec_policy/draft/rule_sets/152/sec_rules/124",
  "created_at": "2016-07-18T23:41:06.092Z",
  "updated_at": "2016-07-18T23:41:06.092Z",
  "deleted_at": null,
  "created_by": {"href": "/users/8"},
  "updated_by": {"href": "/users/8"},
  "deleted_by": null,
  "description": null,
  "enabled": true,
  "providers": [
    {"virtual_server": {"href": "/orgs/1/sec_policy/draft/virtual_servers/f97fc590-2761-422f-a8b2-8227db37e2c1"}}
  ],
  "consumers": [
    {"label": {"href": "/orgs/1/labels/1"}}
  ],
  "consuming_security_principals": [],
  "ingress_services": [],
  "resolve_labels_as": {
    "providers": ["virtual_services"],

```

```

    "consumers": ["workloads"]
  },
  "sec_connect": false,
  "unscoped_consumers": false
},
{
.....
}

```

Curl Command to Get a Rule

```

curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/1/sec_
policy/draft/rule_sets/152/sec_rules/128 -H "Accept: application/json" -u
$KEY:$TOKEN

```

Create Rules

This API allows you to create one or more rules inside a specific ruleset.

URI to Create a Rule

```

POST [api_version][rule_set_href]/sec_rules



```

Request Properties

Property	Description	Type	Required
enabled	Indicates if the rule is enabled or disabled.	Boolean	Yes
providers	Entities that can be used as a provider in a rule, each of which is defined in JSON by its HREF: <ul style="list-style-type: none"> • label • label_group • workload • virtual_server • ip_list 	String	Yes, at least one
consumers	Entities that can be used as a consumer in a rule, each of which is defined in JSON by its HREF:	String	Yes, at least one

Property	Description	Type	Required
	<ul style="list-style-type: none"> • label • label group • workload • ip_list 		
ingress_services	<p>The rule allows connections to the services specified in <code>ingress_services</code>, subject to the value of <code>resolve_labels_as</code>.</p> <p>These parameters work together as follows:</p> <p>If the providers side of <code>resolve_labels_as</code> is set to <code>["workloads"]</code>, then <code>ingress_services</code> contains an array of service HREFs.</p> <p>For example: <code>"ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/20"}]</code></p> <p>If the providers side of <code>resolve_labels_as</code> is set to <code>["virtual_services"]</code>, then <code>ingress_services</code> must contain an empty array.</p> <p>Connections are allowed to the services defined by the matching virtual services.</p> <p>For example: <code>"ingress_services": []</code></p> <p>Finally, if the providers side of <code>resolve_labels_as</code> is set to <code>["workloads", "virtual_services"]</code>, then <code>ingress_services</code> contains an array of service HREFs. When matching a workload, the specified services are allowed by the rule.</p> <p>When matching a virtual service, the services specified in <code>ingress_services</code> are ignored, and the services defined by the virtual service is allowed by the rule.</p>	[String]	Yes
resolve_labels_as	This is a hash with two keys: providers and con-	[String]	Yes

Property	Description	Type	Required
	<p>sumers.</p> <p>The value of each key is an array of strings. Valid strings are "Workloads" and "virtual_services", and the array can contain one or both of these. For each side, if the value is ["workloads"], the rule applies only to workloads that match the rule. If the value is ["virtual_services"], the rule applies only to the matching virtual services. If the value is ["workloads", "virtual_services"] then the rule applies to matching objects of either type. On the providers side, see <code>ingress_services</code> for specific requirements related to the use of either "workloads" or "virtual_services".</p>		
<code>sec_connect</code>	<p>If set to true, then the rule will use SecureConnect IPsec encryption for all traffic allowed by the rule.</p>	Boolean	No
<code>stateless</code>	<p>If set to true, then the rule's packet filtering is stateless. This means that the VEN will instruct the host firewall to not maintain persistent connections for a session. This type of rule is typically used for datacenter "core services" such as DNS and NTP. You can only create a total of 100 stateless rules in your PCE. If you need more than 100 stateless rules in your Illumio policy, contact your Illumio Professional Services Representative for more information.</p>	Boolean	No

Property	Description	Type	Required
	 <p>NOTE: This property has an API exposure level of Public Experimental, which means it is not intended for production use and might change in future releases. For more information, see API Classification and Version.</p>		
machine_auth	<p>If set to true, then machine authentication is used for the rule, meaning that any hosts defined in the rule have been configured for PKI-based machine authentication. Before using this property, your PCE must already be configured for machine authentication. See the PCE Administration Guide for information on configuring machine authentication for the PCE.</p>  <p>NOTE: This property has an API exposure level of Public Experimental, which means it is not intended for production use and might change in future releases. For more information, see API Classification and Version.</p>	Boolean	No
consuming_security_principals	<p>This property is for internal purposes only to enable the PCE to provide Adaptive User Segmentation (AUS) rules. You can ignore and not set this property.</p>	N/A	N/A

Property	Description	Type	Required
unscoped_consumers	Determines if the rule type is intra-scope or extra scope: <ul style="list-style-type: none"> When a rule has <code>unscoped_consumers: false</code>, this defines an intra-scope rule, which means both its providers and consumers are bound by the ruleset scope. When a rule has <code>unscoped_consumers: true</code>, this defines an extra-scope rule, which means its providers are bound by the ruleset scope, but the consumer is <i>not</i> bound by the ruleset scope. 	Boolean	No

Example Payload

This example shows how to construct both intra-scope rules (listed in the JSON as `"unscoped_consumers": false`), and extra-scope rules (listed as `"unscoped_consumers": true`), as well as different types of rule providers and consumers. For information on custom iptables rules, see [Custom iptables Rules](#).

```
{
  "href": "/orgs/1/sec_policy/draft/rule_sets/152/sec_rules/124",

  /* These first two rules are intra-scope rules, *
  /* which consist of basic label to label rules. */

  "enabled": true,
  "providers": [
    {"label": {"href": "/orgs/1/labels/2"}}
  ],
  "consumers": [
    {"label": {"href": "/orgs/1/labels/1"}}
  ],
  "consuming_security_principals": [],
  "ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/20"}],
  "resolve_labels_as": {
```

```

        "providers": ["workloads"],
        "consumers": ["workloads"]
    },

    "sec_connect": true,
    "unscoped_consumers": false
}, /* This rule illustrates using multiple labels for the providers field,
    and an IP List for the consumer field. */
{
    "enabled": true,
    "providers": [
        {"label": {"href": "/orgs/1/labels/3"}},
        {"label": {"href": "/orgs/1/labels/1"}}
    ],
    "consumers": [
        {"ip_list": {"href": "/orgs/1/sec_policy/draft/ip_lists/1"}}
    ],
    "consuming_security_principals": [],
    "ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/28"}],
    "resolve_labels_as": {
        "providers": ["workloads"],
        "consumers": ["workloads"]
    },
    "sec_connect": false,
    "unscoped_consumers": false
},
    /* This rule shows an example of using "all workloads" in a rule. Since this
    is
        an intra-scope rule ("unscoped_consumers": false), "all workloads" here
    means
        all workloads that fall under the current ruleset scopes.*/
{
    "enabled": true,
    "providers": [
        {"ip_list": {"href": "/orgs/1/sec_policy/draft/ip_lists/1"}}
    ],
    "consumers": [
        {"actors": "ams"}
    ],
    "consuming_

```

```
security_principals": [],
  "ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/32"}],
  "resolve_labels_as": {
    "providers": ["workloads"],
    "consumers": ["workloads"]
  },
  "sec_connect": false,
  "unscoped_consumers": false
},

/* The next two rules are extra-scope rules. Notice how in both rules,
"unscoped_consumers": true.
Note too how both intra- and extra-scope rules are placed in the "rules" array
(the only distinction
is the "unscoped_consumers" field) */
/* This is an example of an extra scope rule between labels. Because
the consumers are unscoped, we can fully specify the label set we want, which
in this case is one
each of the Role, Application, Environment, and Location labels */

{
  "enabled": true,
  "providers": [
    {"label": { "href": "/orgs/1/labels/2" }}
  ],
  "consumers": [
    {"label": { "href": "/orgs/1/labels/1" }},
    {"label": { "href": "/orgs/1/labels/24" }},
    {"label": { "href": "/orgs/1/labels/27" }}
  ],
  "consuming_security_principals": [],
  "ingress_services": [{"href": "/orgs/1/sec_policy/draft/services/30"}],
  "resolve_labels_as": {
    "providers": ["workloads"],
    "consumers": ["workloads"]
  },
  "sec_connect": false,
  "unscoped_consumers": true
}
```

```

}

/* This example illustrates an extra-scope rule with an IP list and "all
workloads". In this case,
because we have unscoped consumers, the "all workloads" in consumers actually
means ALL workloads,
not just the ones bound by the ruleset scope as with the intra-scope rules.*/

.....
}

```

Curl Command to Create Rule

```

curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/3/sec_
policy/draft/rule_sets/152/sec_rules -H "Content-Type: application/json" -u
$KEY:$TOKEN -d '{"rules":[{"enabled":true,"providers":[{"label":
{"href":"/orgs/1/labels/2"}}], "consumers":[{"label":
{"href":"/orgs/1/labels/1"}}], "consuming_security_principals":[], "ingress_
services":[{"href":"/orgs/1/sec_policy/draft/services/20"}], "resolve_labels_as":
{"providers":["workloads"], "consumers":["workloads"]}, "sec_
connect":false, "unscoped_consumers":false}, {"enabled":true, "providers":[{"label":
{"href":"/orgs/1/labels/3"}}, {"label":{"href":"/orgs/1/labels/1"}}], "consumers":
[{"ip_list":{"href":"/orgs/1/sec_policy/draft/ip_lists/1"}], "consuming_security_
principals":[], "ingress_services":[{"href":"/orgs/1/sec_
policy/draft/services/28"}], "resolve_labels_as":{"providers":
["workloads"], "consumers":["workloads"]}, "sec_connect":false, "unscoped_
consumers":false}, {"enabled":true, "providers":[{"ip_list":{"href":"/orgs/1/sec_
policy/draft/ip_lists/1"}}], "consumers":[{"actors":"ams"}], "consuming_security_
principals":[], "ingress_services":[{"href":"/orgs/1/sec_
policy/draft/services/32"}], "resolve_labels_as":{"providers":
["workloads"], "consumers":["workloads"]}, "sec_connect":false, "unscoped_
consumers":false}, {"enabled":true, "providers":[{"label":
{"href":"/orgs/1/labels/2"}}], "consumers":[{"label":{"href":"/orgs/1/labels/1"}},
{"label":{"href":"/orgs/1/labels/24"}}, {"label":{"href":"/orgs/1/labels/27"}},
{"label":{"href":"/orgs/1/labels/21"}}], "consuming_security_principals":
[], "ingress_services":[{"href":"/orgs/1/sec_policy/draft/services/30"}], "resolve_
labels_as":{"providers":["workloads"], "consumers":["workloads"]}, "sec_
connect":false, "unscoped_consumers":true}, {"enabled":true, "providers":[{"ip_list":
{"href":"/orgs/1/sec_policy/draft/ip_lists/1"}}], "consumers":

```

```
[{"actors":"ams"},"consuming_security_principals":[],"ingress_services":  
[{"href":"/orgs/1/sec_policy/draft/services/1"}],"resolve_labels_as":{"providers":  
["workloads"],"consumers":["workloads"]},"sec_connect":false,"unscoped_  
consumers":true}]}'
```

Update Rules

This API updates an individual rule inside a ruleset.

URI to Update Rules

```
PUT [api_version][sec_rule_href]
```

The request body and JSON payload is the same as that for [Create Rules](#).

Delete a Rule

This API deletes an individual rule inside a ruleset.

URI to Delete a Rule

```
DELETE [api_version][sec_rule_href]
```

Curl Command to Delete Rule

The curl command for deleting a rule can be structured as follows:

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/sec_  
policy/draft/rule_sets/152/sec_rules/124 -H "Accept: application/json" -u  
$KEY:$TOKEN
```

Rule Search

This Public Experimental method searches for rules across all rulesets. This method is especially useful when your organization has large numbers of rules organized in rulesets. For example, your organization has 192,000 rules organized across 650 rulesets and you needed to know how many rules applied for SNMP (UDP 161). You can't easily find this information without using this method.


NOTE:

Rule search concurrent requests are now increased to 12 searches on 2x2s and 4x2s.

URI to Search for Rules

```
POST api/[api_version]/orgs/:xorg_id/sec_policy/:pversion/rule_search
```

Attributes for Rule Search

You can search for workloads and IP lists by href. The `ingress_services` field accepts either an HREF or an object containing port/protocol/process name/service name, but not `service_ports` or `windows_services` sub-resource.

To search by providers and consumers, you can using the following attributes:

Actor Name	Actor Value Type	Required Keys	Providers	Consumers
actors	String	N/A	True	True
labels	JSON Object	HREF	True	True
label_group	JSON Object	HREF	True	True
workload	JSON Object	HREF	True	True
virtual_service	JSON Object	HREF	True	True
virtual_server	JSON Object	HREF	True	False
ip_list	JSON Object	HREF	True	True

Request Properties

Property	Description	Type
enabled	Whether the rule is enabled or disabled. Returns all the rules that are enabled.	Boolean
description	Description of the rule; can search based on a partial text match. Returns all the rules with a string in their description field (case insensitive partial match).	String
ingress_services	Whether the rule allows connections to services specified in <code>ingress_services</code> . When searching by <code>ingress_services</code> : <ul style="list-style-type: none"> Matches rules that contain all the services that are referenced in the query. Matches rules that contain either services that match the ports/port ranges in the query or uses those ports/port ranges directly. 	Object or null

Property	Description	Type
	<p>Windows services can be searched for implicitly by specifying a <code>service_name</code> or <code>process_name</code> field as one of the objects in the <code>ingress_services</code>.</p> <p>Can be one of two values:</p> <ul style="list-style-type: none"> • HREF • JSON object containing one or more of <code>port</code>, <code>to_port</code>, <code>protocol</code>, <code>service_name</code>, and <code>process_name</code> 	
<code>sec_connect</code>	Whether a secure connection is established in the rule.	Boolean
<code>machine_auth</code>	Whether machine authentication is enabled in the rule.	Boolean
<code>stateless</code>	Whether statelessness is enabled in the rule.	Boolean
<code>unscoped_consumers</code>	<p>Whether the rule type is intra-scope or extra-scope:</p> <ul style="list-style-type: none"> • <code>false</code>: An intra-scope rule • <code>true</code>: An extra-scope rule 	Boolean
<code>update_type</code>	<p>Type of update for the rule. Returns rules with a specific update flag.</p> <p>The string for <code>update_type</code> can include:</p> <ul style="list-style-type: none"> • <code>create</code> • <code>update</code> • <code>delete</code> 	String
<code>providers</code>	<p>Search for rule actors present in providers. Entities that can be used as a provider in a rule, each of which is defined in JSON by its HREF:</p> <ul style="list-style-type: none"> • <code>actors: ams</code> (All Workloads) • <code>label</code> • <code>label group</code> • <code>Workload</code> • <code>Virtual service</code> • <code>virtual server</code> • <code>IP list</code> 	String
<code>consumers</code>	<p>Search for rule actors present in consumers. Entities that can be used as a consumer in a rule, each of which is defined in JSON by its HREF:</p>	String

Property	Description	Type
	<ul style="list-style-type: none"> actors: ams (All Workloads) label label group Workload Virtual service IP list 	

Curl Command Examples for Rule Search

```
$ curl -u API_ID:API_SECRET -X POST -H 'Content-Type: application/json' -d '{
  "providers": [{"label": {"href": "/orgs/1/labels/2"}}, {"consumers": [{"label":
  {"href": "/orgs/1/labels/1"}]}]'https://dev6.ilabs.io:8443/api/v2/orgs/1/sec_
policy/draft/rule_search
```

```
$ curl -u API_ID:API_SECRET -X POST -H 'Content-Type: application/json' -d '{
  "providers": [{"workload": {"href": "/orgs/1/workloads/4ce873d3-2e5d-4f06-82f5-
  4b1e0ec9ceb2"}]}]'https://dev6.ilabs.io:8443/api/v2/orgs/1/sec_policy/draft/rule_
search
```

```
$ curl -u API_ID:API_SECRET -X POST -H 'Content-Type: application/json' -d '{
  "ingress_services": [{"href": "/orgs/1/sec_
  policy/draft/services/1"}]}]'https://dev6.ilabs.io:8443/api/v2/orgs/1/sec_
policy/draft/rule_search
```

```
$ curl -u API_ID:API_SECRET -X POST -H 'Content-Type: application/json' -d '{
  "ingress_services": [{"port": 11000, "to_port": 12000, "proto":
  6}]}]'https://dev6.ilabs.io:8443/api/v2/orgs/1/sec_policy/draft/rule_search
```

Custom iptables Rules

This Public Stable API allows you to leverage preexisting iptables rules on Linux workloads and add them as rules to rulesets.

You can use the rules API to create custom iptables rules in situations where your Linux workloads have preexisting iptables rules configured that you would like to keep in addition to rules you create using Illumio Core.

If you configured iptables on Linux workloads before using Illumio Core, when you pair a workload, the VEN assumes control of the iptables to enact policy and disables any pre-programmed iptables. To solve this, you can use the Rules API to leverage your own iptables rule configurations in a ruleset.

Terminology: Custom iptables Rules

These terms clarify the relationship between your iptables rules and Illumio Core rules:

- **iptables:** Linux host configuration before the VEN is installed
- **Rules:** Configurations in the PCE that define the allowed communication between two or more workloads or other entities (IP lists, labels representing multiple workloads, and label groups)
- **Custom iptables rules:** PCE rules that leverage your iptables rule configurations that get programmed on your workloads by the VEN and managed by the PCE

How Custom iptables Rules Work

Custom iptables rules in the PCE consist of a list of predefined iptables statements and the entities that receive the rule definitions. Each rule can have a list of iptables configurations, which allows you to group a sequence of rules for a specific function. Custom iptables rules are programmed after the Illumio PCE generates the iptables rules and they are provisioned.

Before custom iptables rules are sent to the VEN, they are checked for any unsupported tokens (such as names of firewall chains already in use by Illumio, matching against IP sets, and semicolons). If an unsupported token is included, the rule cannot be saved or provisioned.

If the VEN fails to apply a custom iptables rule because of a missing package or an incorrectly formatted rule:

- The error is reported to the PCE and is logged as two audit events: “Firewall config failure” (`fw_config_failure`) and “Failed to apply policy changes” (`policy_deploy_failed`).
- The error is displayed in the VEN health status.
- The new policy is not used and the last known successful policy is used instead.

For policy distribution and enforcement, the VEN creates a custom chain that contains the rules for each table or chain in the iptables. Each custom chain is appended to the end of its corresponding chain in the correct table. When the VEN requests the policy, the `iptables` command is sent, including where the chain should be placed.

For security reasons, custom iptables rules only support rules in the `mangle`, `nat`, and `filter` tables.

The following table describes the permitted actions for each iptables type:

Table Name	Chain Names	Custom Rules
<code>raw</code>	<code>prerouting</code> , <code>output</code>	No
<code>mangle</code>	<code>prerouting</code> , <code>input</code> , <code>output</code> , <code>forward</code> , <code>postrouting</code>	Yes
<code>nat</code>	<code>prerouting</code> , <code>output</code> , <code>postrouting</code>	Yes
<code>filter</code>	<code>input</code> , <code>output</code> , <code>forward</code>	Yes
<code>security</code>	<code>input</code> , <code>output</code> , <code>forward</code>	No

Create a Custom iptables Rule

This method allows you to create a rule that can contain custom iptables.

URI to Create a Custom iptables Rule

```
POST [api_version/[rule_set_href]/sec_rules
```

Request Parameters

Resource	Description	Type	Required
<code>name</code>	Ruleset name (must be unique)	String	Yes
<code>scopes</code>	Scope for ruleset, which consists of a list of labels, with each list having at least one application, environment, and/or location label	Array	Yes
<code>external_data_set</code>	External data set identifier	String	No
<code>external_data_reference</code>	External data reference identifier	String	No
<code>enabled</code>	Whether the ruleset is enabled or not	Boolean	Yes
<code>rules</code>	Standard (non-iptables) rules	String	Yes
<code>iptables_rules</code>	Rules that use iptables (see following table for properties)	String	Yes

Custom iptables_rules Properties

Property	Description	Type	Required
<code>enabled</code>	Whether the rule is currently enabled	Enum	Yes
<code>ip_version</code>	Whether IPv4 or IPv6 is used	String	Yes

Property	Description	Type	Required
description	Description of ruleset	String	No
actors	Entities that receive the ruleset.	String	Yes
statements	Rules for iptables (table, chain name, and parameters), which consist of the following elements: <ul style="list-style-type: none"> • <code>table_name</code>: Name of iptables table, which is <code>nat</code>, <code>mangle</code>, or <code>filter</code> • <code>chain_name</code>: Name of iptables chain, which is <code>prerouting</code>, <code>input</code>, <code>output</code>, <code>forward</code>, or <code>postrouting</code> • <code>parameters</code>: Remaining iptables rules (excluding table name and chain name) 	String	Yes

For more information on rules, see [Rulesets](#).

Request Body

In this example, a ruleset named `test ipt rs` is created that contains two iptables rules.



NOTE:
Each iptables rule can contain multiple statements.

```
{
  "name": "test ipt rs",
  "enabled": true,
  "scopes": [
    [
      { "label": { "href": "/orgs/1/labels/24" } },
      { "label": { "href": "/orgs/1/labels/27" } },
      { "label": { "href": "/orgs/1/labels/21" } }
    ],
  ],
  "ip_tables_rules": [
    {
      "enabled": true,
      "actors": [{"label": { "href": "/orgs/1/labels/11" }}],
      "statements": [
        {
```

```

        "table_name": "mangle",
        "chain_name": "PREROUTING",
        "parameters": "-i eth0 -p tcp --dport 2222 -j MARK --set-mark
2222"
    },
    {
        "table_name": "nat",
        "chain_name": "PREROUTING",
        "parameters": "-i eth0 -p tcp -m mark --mark 2222 -j REDIRECT
--to-port 3333"
    },
    {
        "table_name": "filter",
        "chain_name": "INPUT",
        "parameters": "-i eth0 -p tcp -m mark --mark 2222 -j ACCEPT"
    }
],
"ip_version": "4"
},
{
    "enabled": true,
    "actors": [{ "actors": "ams" }],
    "statements": [
        {
            "table_name": "nat",
            "chain_name": "POSTROUTING",
            "parameters": "-o eth1 -s 192.0.2.10! -d 198.51.100.0/24 -j
MASQUERADE"
        }
    ],
    "ip_version": "4"
}
]
}

```

Curl Command to Create Custom iptables Rule

```

curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/draft/rule_sets -H "Content-Type:application/json" -u $KEY:$TOKEN-d '

```

```
{
  "name": "test_ipt_rs",
  "enabled": true,
  "scopes": [
    [
      [
        {
          "enabled": true,
          "actors": [
            {
              "label": {
                "href": "/orgs/1/labels/11"
              }
            }
          ],
          "statements": [
            {
              "table_name": "mangle",
              "chain_name": "PREROUTING",
              "parameters": "-i eth0 -p tcp --dport 2222 -j MARK --set-mark 2222"
            },
            {
              "table_name": "nat",
              "chain_name": "PREROUTING",
              "parameters": "-i eth0 -p tcp -m mark --mark 2222 -j REDIRECT --to-port 3333"
            },
            {
              "table_name": "filter",
              "chain_name": "INPUT",
              "parameters": "-i eth0 -p tcp -m mark --mark 2222 -j ACCEPT"
            }
          ],
          "ip_version": "4"
        },
        {
          "enabled": true,
          "actors": [
            {
              "actors": "ams"
            }
          ],
          "statements": [
            {
              "table_name": "nat",
              "chain_name": "POSTROUTING",
              "parameters": "-o eth1 -s 10.0.0.2 ! -d 172.17.0.0/16 -j MASQUERADE"
            }
          ],
          "ip_version": "4"
        }
      ]
    ]
  ]
}
```

Response Body

Property	Description	Type
href	Identifier for the resource	String

Response

```
{
  "href": "/orgs/1/sec_policy/draft/rule_sets/17",
  "created_at": "2016-02-24T23:19:01.020Z",
  "updated_at": "2016-02-24T23:19:01.020Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/1"
  },
  "updated_by": {
    "href": "/users/1"
  },
  "deleted_by": null,
  "name": "test_ipt_rs",
  "description": null,
  "enabled": true,
  "scopes": [
    [
      [
        {
          "label": {
            "href": "/orgs/1/labels/24"
          }
        },
        {
          "label": {
            "href": "/orgs/1/labels/27"
          }
        },
        {
          "label": {
            "href": "/orgs/1/labels/21"
          }
        }
      ]
    ]
  ]
}
```

```
{ "label": { "href": "/orgs/1/labels/15" } },
{ "label": { "href": "/orgs/1/labels/16" } },
{ "label": { "href": "/orgs/1/labels/17" } }
]
],
],
"rules": [],
"ip_tables_rules": [
  {
    "href": "/orgs/1/sec_policy/draft/rule_sets/17/ip_tables_rules/20",
    "created_at": "2016-02-24T23:19:01.280Z",
    "updated_at": "2016-02-24T23:19:01.280Z",
    "deleted_at": null,
    "created_by": {
      "href": "/users/1"
    },
    "updated_by": {
      "href": "/users/1"
    },
    "deleted_by": null,
    "description": null,
    "enabled": true,
    "actors": [
      {
        "actors": "ams"
      }
    ],
    "ip_version": "4",
    "statements": [
      {
        "table_name": "nat",
        "chain_name": "POSTROUTING",
        "parameters": "-o eth1 -s 192.0.2.0 ! -d 198.51.100.0/24 -j MASQUERADE"
      }
    ]
  }
],
{
  "href": "/orgs/1/sec_policy/draft/rule_sets/17/ip_tables_rules/18",
```

```
"created_at": "2016-02-24T23:19:01.229Z",
"updated_at": "2016-02-24T23:19:01.229Z",
"deleted_at": null,
"created_by": {
  "href": "/users/1"
},
"updated_by": {
  "href": "/users/1"
},
"deleted_by": null,
"description": null,
"enabled": true,
"actors": [
  {
    "label": {
      "href": "/orgs/1/labels/11",
      "key": "loc",
      "value": "test"
    }
  }
],
"ip_version": "4",
"statements": [
  {
    "table_name": "filter",
    "chain_name": "INPUT",
    "parameters": "-i eth0 -p tcp -m mark --mark 2222 -j ACCEPT"
  },
  {
    "table_name": "nat",
    "chain_name": "PREROUTING",
    "parameters": "-i eth0 -p tcp -m mark --mark 2222 -j REDIRECT --to-port
3333"
  },
  {
    "table_name": "mangle",
    "chain_name": "PREROUTING",
    "parameters": "-i eth0 -p tcp --dport 2222 -j MARK --set-mark 2222"
```



```
    }  
  ]  
}  
]  
}
```

Machine Authentication

This Public Experimental API allows you to configure unmanaged workloads and rules for machine authentication in case you configured the PCE to use machine authentication.

Before you start writing rules, you need to complete the following tasks:

- Configure an unmanaged (no VEN) workload that you want to use machine authentication on with the client certificate X.509 Subject distinguished name (`distinguished_name`) issued from the CA. If you are using machine authentication with managed workloads (with VENs installed), you do not need to set this property.
- Configure rules for machine authentication by setting the `machine_auth` flag to true on each rule. You can also optionally set SecureConnect (`sec_connect`) if you want the traffic data to be encrypted using IPsec.

Once you have done these two tasks, you can use these unmanaged workloads in machine authentication-based rules.

Configure Machine Authentication

The machine authentication workload property for the certificate distinguished name is required for those hosts or systems where you have not installed a VEN, such a laptop or other server whose IP address is unknown or changes often.

You can set the `distinguished_name` when you first create (POST) the unmanaged workload, which is passed in the JSON request payload.



NOTE:

For information on how to create an unmanaged workload, see [Create an Unmanaged Workload](#).

URI to Configure Machine Authentication on an Unmanaged Workload

Use this URI to configure machine authentication when you create a new unmanaged workload:

```
POST [api_version][org_href]/workloads
```

If you want to enable machine authentication on an existing unmanaged workload, you need to know the workload HREF, which can be obtained from the command GET on a collection of Workloads.

The workload HREF is highlighted in blue:

```
/orgs/7/workloads/XXXXXXX-9611-44aa-ae06-fXXX8903db65
```

Use this URI to configure machine authentication for an existing unmanaged workload:

```
PUT [api_version][workload_href]
```

Request Property

Property	Description
distinguished_name	The X.509 Subject distinguished name, used if you want this unmanaged workload to use machine authentication when communicating with other hosts.

Request Body

```
{  
  "distinguished_name": "CN=ACCVRAIZ1, OU=PKIACCV, O=ACCV, C=ES"  
}
```

Curl Command Enable Machine Authentication

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/7/workloads/XXXXXXX-9611-44aa-ae06-fXXX8903db65 -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"distinguished_name": "CN=ACCVRAIZ1, OU=PKIACCV, O=ACCV, C=ES"}'
```

Configure Machine Authentication on Rule

For a rule to use machine authentication, you need to configure it on the rule when you create or update it.

URI to Configure Machine Authentication for a Rule

Use this URI to configure machine authentication for a new rule:

```
POST [api_version][rule_set_href]/sec_rules
```

If you want to enable machine authentication on an existing rule, you need to know the HREF of the rule. For example:

```
/orgs/3/sec_policy/draft/rule_sets/152/sec_rules/124
```

Use this URI to configure machine authentication for an existing rule:

```
PUT [api_version][sec_rule_href]
```

Request Properties

Property	Description
machine_auth	Optional boolean flag to enable machine authentication for the rule. When set to true, machine authentication is enabled for the rule.
sec_connect	Optional boolean flag to enable SecureConnect (host-to-host traffic encryption) for the rule.

Request Body

This example shows the JSON payload for updating a rule to enable machine authentication, but with SecureConnect disabled.

```
{
  "providers": [{"label": {"href": "/orgs/1/labels/1"}}],
  "sec_connect": false,
  "consumers": [{
    "actors": "ams"
  }],
  "consuming_security_principals": [],
  "unscoped_consumers": false,
  "description": ""
```

```
"ingress_services": [{"proto": 6}],
"resolve_labels_as": {
  "providers": ["workloads"],
  "consumers": ["workloads"]
},
"enabled": true,
"machine_auth": true
}
```

Curl Command to Configure Machine Authentication for Rule

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/1/sec_policy/draft/rule_
sets/152/sec_rules/124 -H "Content-Type:application/json" -u $KEY:$TOKEN -d '
{"providers":[{"label": {"href":"/orgs/1/labels/1"}}], "sec_connect":false,
"consumers":[{"actors":"ams"},"consuming_security_principals":[], "ingress_
services": [{"proto": 6}], unscoped_consumers":false, "description":", "resolve_
labels_as":{"providers":["workloads"],"consumers":
["workloads"]},"enabled":true,"machine_auth":true}'
```

Security Policy Objects

This chapter contains the following topics:

Overview of Security Policy Objects	230
Labels	230
Label Groups	237
Services	243
Virtual Services and Service Bindings	251
IP Lists	267
Virtual Servers and Load Balancers	274
Security Principals	280

The security policy in Illumio represents a configurable set of rules that protects network assets from threats and disruptions and secures communications between workloads.

The PCE contains security objects, such as IP lists, labels, label groups, and services to help you write your security policy. These objects define version, modifications, dependencies, changes, and whether a policy can be reverted.

In the Illumio's label-based system, the rules you write don't require the use of an IP address or subnet, and you can control the range of your policy by using labels. Use label groups to write rules more efficiently if the same labels are used repeatedly in rulesets.

Overview of Security Policy Objects

Security policy objects contain information about policy versions, modifications, whether it is still pending, and can be reverted, policy dependencies, and policy changes.

Active vs. Draft

This Public Stable API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, fire-wall settings, SecureConnect gateways, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID if the security policy.

Labels

This Public Stable API gets, creates, updates, and deletes labels.

Labels API Methods

Functionality	HTTP	URI
Get a collection of labels	GET	[api_version][org_href]/labels
Get an individual label	GET	[api_version][label_href]
Create a label	POST	[api_version][org_href]/labels
Update a label	PUT	[api_version][label_href]
Delete a label	DELETE	[api_version][label_href]

Get Labels

This API returns all labels in an organization or a single label. When you get labels, they are returned in the form of an HREF path property, for example: `"/orgs/2/labels/1662"`

By default, the maximum number returned on a GET collection of labels is 500. To return more than 500 labels, use an [Asynchronous GET Collection](#).


NOTE:

GET returns any label that contains a match, as opposed to an exact match. For example, a GET request for labels with value=APP could return APP, WEB-APP, WEBAPP.

URI to Get Collection of Labels

```
GET [api_version][org_href]/labels
```

URI to Get an Individual Label

```
GET [api_version][label_href]
```

Query Parameters

Parameter	Description	Type
key	This string indicates the type of label you want to get; includes these four types: app, env, role, loc	String
usage	Indicate label usage, including if the label is currently being used in an RBAC scope for user permissions, if the label is being applied to a virtual service, pairing profile, virtual server or ruleset, selective enforcement, and if the label belongs to a label group.	Boolean
include_deleted	Include records of all labels that have been deleted.	Boolean
external_data_set	The data source from which the resource originates. For example, if this label information is stored in an external database.	String
external_data_reference	A unique identifier within the external data source, used only if this label information is stored in an external database.	String
value	The value of the label to return.	
max_results	The maximum number of results to return when using the GET method. The maximum limit for returned labels is 500. To return more than 500 labels, use an Asynchronous	Integer

Parameter	Description	Type
	GET Collection.	

Query Parameters to Get an Individual Label

Parameter	Description	Type
usage	Indicate label usage, including if the label is currently used in an RBAC scope for user permissions, if the label is applied to a workload, virtual service, pairing profile, selective enforcement, virtual server, or rule-set, and if the label belongs to a label group.	Boolean

Curl Command to Get Collection of Labels

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/labels -H "Accept: application/json" -u $KEY:$TOKEN
```

Response Body

In the response body, each label returned is identified as an HREF, for example: `"/orgs/2/labels/1662"`

For example:

```
{
  href: "/orgs/2/labels/1662"
  key: "env"
  value: "Prod"
  created_at: "2014-01-22T18:24:33Z"
  updated_at: "2014-01-22T18:24:40Z"
  created_by: {
    href: "/users/9"
  }
  updated_by: {
    href: "/users/9"
  }
}
{
  href: "/orgs/2/labels/1128"
  key: "role"
```



```
    value: "DB"
    created_at: "2014-01-22T18:24:53Z"
    updated_at: "2014-01-22T18:24:59Z"
    created_by: {
      href: "/users/9"
    }
    updated_by: {
      href: "/users/9"
    }
  }
  {
    href: "/orgs/2/labels/1637"
    key: "app"
    value: "Store"
    created_at: "2014-02-24T17:28:43Z"
    updated_at: "2014-02-24T17:28:43Z"
    created_by: {
      href: "/users/9"
    }
    updated_by: {
      href: "/users/9"
    }
  }
  {
    href: "/orgs/2/labels/1638"
    key: "app"
    value: "HRM"
    created_at: "2014-02-24T17:28:57Z"
    updated_at: "2014-02-24T17:28:57Z"
    created_by: {
      href: "/users/9"
    }
    updated_by: {
      href: "/users/9"
    }
  }
}
```

Curl Command to Get a Label

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/labels/8 -H "Accept: application/json" -u $KEY:$TOKEN
```

Response Body

```
{
  href: "/orgs/2/labels/8"
  key: "env"
  value: "Prod"
  created_at: "2014-01-22T18:24:33Z"
  updated_at: "2014-01-22T18:24:40Z"
  created_by: {
    href: "/users/9"
  }
  updated_by: {
    href: "/users/9"
  }
}
```

Create a Label

This API creates a new label inside an organization for one of the following label types, for which you can provide your own string value:

- **Application** (“app”): The type of application the workload is supporting. For example, HRM, SAP, Finance, Storefront.
- **Role** (“role”): The function of a workload. In a simple two-tier application consisting of a web server and a database server, there are two roles: Web and Database.
- **Environment** (“env”): The stage in the development of the application. For example, production, QA, development, staging.
- **Location** (“loc”): The location of the workload. For example, Germany, US, Europe, Asia; or Rack #3, Rack #4, Rack #5; or data center, AWS-east1, AWS-east2, and so on.

System Default “All” for Labels

The PCE provides built-in environment, application, and location labels that are defined as “All” that create broad policies to cover all applications, all environments, and all locations.

For this reason, you cannot create labels of these types defined as “All Applications,” “All Environments,” or “All Locations” (exactly as written in quotes) in order to prevent confusion for policy writers.

If you attempt to create labels of these types with the exact name as the system defaults (for example, “All Applications”), you receive an HTTP “406 Not Acceptable” error.

Illumio recommends not creating labels with names similar to these default system labels to avoid confusion.

URI to Create a Label

```
POST [api_version][org_href]/labels
```

Request Properties

Property	Description	Type	Required
key	This string indicates the type of label you want to create; includes these four types: app, env, role, loc	String	Yes
value	Value of the label that you are updating for the specified label.	String	Yes
external_data_set	The data source from which the resource originates. For example, if this label information is stored in an external database.	String	No
external_data_reference	A unique identifier within the external data source. For example, if this label information is stored in an external database.	String	No

Example Request Body

```
{
  "key": "role",
  "value": "web"
}
```

Curl Command to Create a Label

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/labels -H "Content-Type: application/json" -u $KEY:$TOKEN -d '{"key": "role", "value": "web"}'
```

Response Body

The created label is in the form of an HREF path property. For example, in the response below, the label is identified as `"/orgs/2/labels/1677"`.

```
{
  href: "/orgs/2/labels/1677"
  key: "role"
  value: "my_web_app"
  created_at: "2014-04-18T19:39:27Z"
  updated_at: "2014-04-18T19:39:27Z"
  created_by: {
    href: "/users/76"
  }
  updated_by: {
    href: "/users/76"
  }
}
```

Update a Label

This API allows you to update a label applied to a workload, given that you have the label HREF, which is returned when you get all labels in an organization. For example: `"/orgs/2/labels/1662"`

URI to Update a Label

```
PUT [api_version][label_href]
```

Request Body

Property	Description	Type
value	Value of the label being updated.	String
external_data_set	The data source from which the resource originates. For example, if this label information is stored in an external database.	String
external_data_reference	A unique identifier within the external data source. For example, if this labels information is stored in an external database.	String

Example Request Body

To update a label definition, the JSON request body can be constructed as follows:

```
{ "value": "db" }
```

Curl Command to Update a Label

```
curl -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/labels/1662 -H "Accept: application/json" -u $KEY:$TOKEN -d '{"value": "db"}
```

Delete a Label

This API deletes a label from an organization using the label HREF, which is returned when you get a collection of labels in an organization. For example: `"/orgs/2/labels/1662"`

URI to Delete a Label

```
DELETE [api_version][label_href]
```

Curl Command to Delete a label

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/labels/1662 -H "Accept: application/json" -u $KEY:$TOKEN
```

Label Groups

This Public Stable API helps you write rules more efficiently if the same labels are used repeatedly in rulesets. When you add labels to a label group, the label group can be

used in a rule or ruleset scope to represent multiple labels. A label group can also be a member (child) of other label groups.

Label Groups API Methods

Functionality	HTTP	URI
Get a collection of label groups	GET	[api_version][org_href]/sec_policy/draft/label_groups
Create a new label group	POST	[api_version][org_href]/sec_policy/draft/label_groups
Get an individual label group	GET	[api_version][label_group_href]
Get an individual label group to see if it is a member of other label groups	GET	[api_version][label_group_href]/member_of
Update an individual label group	PUT	[api_version][label_group_href]
Delete an individual label group	DELETE	[api_version][label_group_href]

Active vs. Draft

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, firewall settings, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID of the security policy.

Get Collection of Label Groups

This method gets all label groups in your organization. Use this to discover the `label_group_id` to GET a specific label group or for POST, PUT, and DELETE operations.

By default, the maximum number returned on a GET collection of label groups is 500. If you want to get more than 500 label groups, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Label Groups


```
GET [api_version][org_href]/sec_policy/draft/label_groups
```

URI to Get an Individual Label

```
GET [api_version][label_group_href]
```

Query Parameters

Use the following optional query parameters to restrict the results of the query when getting a collection of label groups.

Parameter	Description	Type
name	The specific name of a label group to return.	String
description	The description of the label group to return. Partial matches are supported.	String
key	This string indicates the type of label group you want to get, of which there are four types: app, env, role, loc	String
max_results	The maximum number of results you want to return when using the GET method. The maximum limit for returned results is 500. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  NOTE: If this parameter is not specified, or a value greater than 500 is specified, a maximum of 500 results are returned. </div>	Integer
external_data_set	The data source from which the resource originates. For example, if label group information is stored in an external database.	String
external_data_reference	A unique identifier within the external data source. For example, if label group information is stored in an external database.	String

Parameter	Description	Type
usage	Return label usage flags	Boolean

Curl Command to Get Label Groups

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/sec_policy/draft/label_groups -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

When you get a collection of label groups, each label group is identified by an HREF. You need the HREF to update or delete an individual label group using the API.

```
{
  "href": "/orgs/2/sec_policy/draft/label_groups/3307b3d8-2ca2-48f5-877a-03ada95cd6de",
  "created_at": "2015-07-25T00:58:31.046Z",
  "updated_at": "2015-07-25T00:58:31.046Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/3"
  },
  "updated_by": {
    "href": "/users/3"
  },
  "deleted_by": null,
  "name": "AppGroup",
  "description": null,
  "key": "app",
  "labels": [],
  "sub_groups": [
    {
      "href": "/orgs/2/sec_policy/draft/label_groups/9b30081e-e105-44d8-9945-4c8a30dbe849",
      "name": "AppGroup3"
    }
  ]
},
{
  "href": "/orgs/2/sec_policy/draft/label_groups/4c8e3325-c6dd-4dc2-aadc-
```



```
971e9de270e4",
  "created_at": "2015-07-25T00:46:52.552Z",
  "updated_at": "2015-07-25T00:59:00.177Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/3"
  },
  "updated_by": {
    "href": "/users/3"
  },
  "deleted_by": null,
  "name": "AppGroup1",
  "description": null,
  "key": "app",
  "labels": [],
  "sub_groups": [
    {
      "href": "/orgs/2/sec_policy/draft/label_groups/9b30081e-e105-44d8-9945-4c8a30dbe849",
      "name": "AppGroup3"
    }
  ]
},
```

Label Group Belonging to Other Groups

This method determines if an individual label group is a member of other label groups. For example, if one label group is also a “child” of three other label groups, the response to this call returns the three “parent” label groups to which the specified label group belongs.

URI to Check if a Label Group Belongs to Other Label Groups

```
GET [api_version][label_group_href]/member_of
```

Response

If the specified label group does not belong to any other label groups, the call returns an HTTP 200 message. If the specified label group does belong to other label groups, the response lists the parent label groups. For example:

```
[
  {
    "href": "/orgs/7/sec_policy/draft/label_groups/b51c986b-db35-47d4-ab77-
    aae570d1f164",
    "name": "MyLablesUS"
  }
]
```

Update a Label Group

To update an individual label group, use the HREF of the label group, which is obtained from an API call to get a collection of label groups.

URI to Update a Label Group

```
PUT [label_group_href]
```

Request Body

This example request body updates the labels contained within a label group.

```
{
  "labels": [
    { "href": "/orgs/28/labels/1100" },
    { "href": "/orgs/28/labels/1098" },
    { "href": "/orgs/28/labels/1099" },
    { "href": "/orgs/28/labels/1101" }
  ],
  "sub_groups": []
}
```

Curl Command to Update Label Groups

In this example, the label group being updated with the request body from the code example above is identified by the its label group HREF.

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/draft/label_groups/3307b3d8-2ca2-48f5-877a-03ada95cd6de -H "Content-
Type:application/json" -u $KEY:$TOKEN -d '{"labels":
```

```
[{"href":"/orgs/28/labels/1100"}, {"href":"/orgs/28/labels/1098"}, {"href":"/orgs/28/labels/1099"}, {"href":"/orgs/28/labels/1101"}], "sub_groups": []}'
```

Delete a Label Group

To delete an individual label group, specify the HREF of the label group you want to delete, which is obtained from an API call to get a collection of label groups.

URI to Delete a Label Group

```
DELETE [api_version][label_group_href]
```

Curl Command to Delete a Label Group

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/draft/label_groups/3307b3d8-2ca2-48f5-877a-03ada95cd6de -u $KEY:$TOKEN
```

Services

This Public Stable API gets, creates, updates, or deletes services. To write services they must be in the “draft” state, which means they have not been provisioned. To provision changes made to services, use the Security Policy API.

Services API Methods

Functionality	HTTP	URI
Get a collection of services	GET	[api_version][org_href]/sec_policy/{pversion}/services
Get an individual service	GET	[api_version][org_href]/sec_policy/{pversion}/services/service_id
Create a new service	POST	[api_version][org_href]/sec_policy/draft/services
Update an individual service	PUT	[api_version][org_href]/sec_policy/draft/services/service_id
Delete an individual service	DELETE	[api_version][org_href]/sec_policy/draft/services/service_id

Active vs. Draft

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, firewall settings, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID of the security policy.

Get Services

This API gets all the services in your organization that are in the “draft” policy state (not yet provisioned).

By default, the maximum number returned on a GET collection of services is 500. To get more than 500 services, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Services

```
GET [api_version][org_href]/sec_policy/draft/services
```

URI to Get an Individual Service


```
GET [api_version][service_href]
```

Query Parameters

Use the following optional query parameters to restrict the results of the query.

Parameter	Description	Type
<code>external_data_set</code>	The data source from which the resource originates. For example, if service information is stored in an external database.	String
<code>external_data_reference</code>	A unique identifier within the external data source. For example, if service information is stored in an external database.	String
<code>name</code>	Name of service on which to filter. This parameter supports partial matches.	String
<code>description</code>	Description of the service on which to filter. This parameter supports partial matches.	String
<code>port</code>	Port on which to filter. This parameter supports partial	String

Parameter	Description	Type
	matches. The range is from 1 to 65535. Enter -1 for any port.	
proto	Protocol on which to filter. This parameter supports partial matches.	Integer
max_results	The maximum number of results to return using GET. The maximum limit for returned services is 500.	Integer



NOTE:
If this parameter is not specified, or a value greater than 500 is specified, a maximum of 500 results are returned.
To get more than 500 services, use an [Asynchronous GET Collection](#).

Curl Command to Get All Services

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/draft/services -H "Accept: application/json" -u $KEY:$TOKEN
```

Curl Example to Get a Service

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/draft/services/91 -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

Each individual service returned is identified by a service HREF. To GET, PUT, or DELETE an individual service, identify the service using its HREF in the API call.

```
{
  "href": "/orgs/2/sec_policy/draft/services/91",
  "created_at": "2015-09-02T08:42:02.299Z",
  "updated_at": "2015-09-02T08:42:02.299Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/4"
  },
}
```

```
"updated_by": {
  "href": "/users/4"
},
"deleted_by": null,
"name": "RabbitMQ",
"description": "RabbitMQ",
"description_url": null,
"process_name": null,
"service_ports": [
  {
    "port": 5672,
    "proto": "tcp"
  }
]
},
{
  "href": "/orgs/2/sec_policy/draft/services/77",
  "created_at": "2015-09-02T08:41:59.921Z",
  "updated_at": "2015-09-02T08:41:59.921Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/4"
  },
  "updated_by": {
    "href": "/users/4"
  },
  "deleted_by": null,
  "name": "PostgreSQL",
  "description": "PostgreSQL",
  "description_url": null,
  "process_name": null,
  "service_ports": [
    {
      "port": 5432,
      "proto": "tcp"
    }
  ]
},
```

```
{
  "href": "/orgs/2/sec_policy/draft/services/79",
  "created_at": "2015-09-02T08:42:00.315Z",
  "updated_at": "2015-09-02T08:42:00.315Z",
  "deleted_at": null,
  "created_by": {
    "href": "/users/4"
  },
  "updated_by": {
    "href": "/users/4"
  },
  "deleted_by": null,
  "name": "Tomcat",
  "description": "Tomcat",
  "description_url": null,
  "process_name": null,
  "service_ports": [
    {
      "port": 8080,
      "proto": "tcp"
    }
  ]
},
{
  "href": "/orgs/7/sec_policy/active/services/878",
  "created_at": "2017-02-10T18:10:50.324Z",
  "updated_at": "2017-02-10T18:10:50.324Z",
  "deleted_at": null,
  "updated_by": null,
  "deleted_by": null,
  "name": "ICMP ECHO",
  "description": null,
  "description_url": null,
  "process_name": null,
  "service_ports": [
    {
      "icmp_type": 8,
      "icmp_code": null,
    }
  ]
}
```

```

        "proto": 1
    },
    {
        "icmp_type": 128,
        "icmp_code": null,
        "proto": 58
    }
]
}
    
```

Create a Service

This method creates an individual service. Once a service is created, it can be used to write rules for a security policy.

URI to Create a Service

```
POST [api_version][org_href]/sec_policy/draft/services
```

Request Properties

Property	Description	Type	Required
name	Name of the service (does not need to be unique).	String	Yes
description	Optional description for the service.	String	No
process_name	Name of the process.	String	No
service_ports			
	id. Process ID of the service.	Integer	No
	port. Port number for the service.	Integer	Yes
	to_port. Destination port for the service.	Integer	Yes
	proto. Transport protocol to be used for the service.	Integer	Yes
windows_services			
	id: Process ID of the Windows service.	Integer	No
	port: Port number for the Windows service.	Integer	Yes
	to_port: Destination port for the Windows service.	Integer	Yes
	proto: Transport protocol used for the Windows service.	Integer	Yes

Property	Description	Type	Required
	service_name: Name of the Windows service.	String	No
	process_name: Name of the running Windows process.	String	No
external_data_set	The data source from which the resource originates. For example, if this service's information is stored in an external database.	String	No
external_data_reference	The external data reference from which the resource originates. For example, if this service's information is stored in an external database.	String	Optional

Example Payload

```
{
  "name": "RDP",
  "description": "Windows Remote Desktop",
  "service_ports": [
    {
      "port": 3389,
      "proto": 6
    }
  ]
}
```

Curl Command to Create Windows Service

This example shows how to create a Windows Remote Desktop (RDP) service.

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/active/services -H "Content-Type:application/json" -u $KEY:$TOKEN -d '
{"name":"RDP", "description":"Windows Remote Desktop","service_ports":
[{"port":3389,"proto":6}]}'
```

Update a Service

In order to update (PUT) an individual service, you need to know the HREF of the service you want to update. A service's HREF is returned when you get a collection of

services from the PCE.

URI to Update an Individual Service

```
PUT [api_version][service_href]
```

Request Body

This example illustrates the request body you can pass to update a service, for example, to change the port used by the Nginx service from its current port number to 8080:

```
{
  "name": "nginx",
  "service_ports": [
    {
      "port": 8080,
      "proto": 6
    }
  ]
}
```

Curl Command to Update Service

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/active/services/79 -H "Content-Type:application/json" -u $KEY:$TOKEN -d '
{"name":"nginx","service_ports":[{"port":8080,"proto":6}]}'
```

Delete a Service

To delete an individual service, use the HREF of the service you want to delete, which is returned when you get a collection of services.

URI to Delete an Individual Service

```
DELETE [api_version][service_href]
```

Curl Command to Delete Service

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/active/services/79 -u $KEY:$TOKEN
```

Virtual Services and Service Bindings

This Public Stable API gives you the ability to write rules on a per-service basis instead of having to write rules that apply to all the services running on a workload. By binding a workload to individual services, you can isolate one or more services running on a workload and create policies specific to those services. By binding services, you have the flexibility to create a finely-grained, highly-segmented security policy.

Once you have created, provisioned, and bound a virtual service to a specific workload, you can use the virtual service in rules. See [Create an Individual Virtual Service](#) and [Create a Service Binding](#) for information.

Virtual Services

Virtual services can consist of either a single service or a collection of explicitly enumerated port/port range and protocol tuples. They can be used directly in a rule as a single entity, or labels that represent multiple virtual services can be used to write rules.

Virtual services are dynamically bound to workloads using service bindings. Create a virtual service, and then use a service binding to bind the specific virtual service to a workload. Rules written using a virtual service only apply to the workload to which the service is bound.

Use virtual services in the following scenarios:

- **Apply Rules to a Single Service**

This scenario represents a service or process on a workload using a name or label. You can write a policy that allows other entities to communicate only with that single service. The policy does not need to change if the service is moved to a different workload or a new set of workloads. Only the workload bindings on the virtual service need to be changed. The PCE dynamically calculates the required rules on the updated workloads to allow this virtual service.

- **Applying Rules to one of the many Virtual Services running on a Workload**

In this case, multiple virtual services are running on the workload, with different labels, and the rule targets a subset of those services. You can write a rule to

allow other entities to communicate only with that specific service. The policy does not need to change if this service is moved to a different workload or a new set of workloads. Only the workload bindings on the virtual service need to be changed. The PCE dynamically calculates the required rules on the updated workloads to allow the virtual service.

Virtual Services API Methods

Functionality	HTTP	URI
Get a collection of virtual services	GET	[api_version][org_href]/sec_policy/:pversion/virtual_services
Get an individual virtual service	GET	[api_version][org_href]/sec_policy/:pversion/virtual_services/virtual_service_id
Create a new virtual service	POST	[api_version][org_href]/sec_policy/draft/virtual_services
Create a collection of virtual services	PUT	[api_version][org_href]/sec_policy/draft/virtual_services/bulk_create
Update a virtual service	PUT	[api_version][org_href]/sec_policy/draft/virtual_services/virtual_service_id
Update a collection of virtual services	PUT	[api_version][org_href]/sec_policy/draft/virtual_services/bulk_update
Delete a virtual service	DELETE	[api_version][org_href]/sec_policy/draft/virtual_services/virtual_service_id

Active vs. Draft Policy Items

Because virtual services are policy items, changes made to them must be provisioned before they can take effect on your policy. Policy items always exist in either a draft (not provisioned) or active (provisioned) state.

Security policy items that must be provisioned to take effect include IP lists, rulesets, rules, services, virtual services, label groups, user groups, virtual servers, and PCE security settings.

For these items, the URL of the API call must include the URI element called :pversion, which can be set to either draft or active when you make the API call.

Depending on the method, the API follows these rules:

- For GET operations — :pversion can be draft or active
- For POST, PUT, DELETE — :pversion can only be draft (you cannot operate on provisioned items)

Get Collection of Virtual Services

Use this method to get a collection of Virtual Services.


URI to Get a Collection of Virtual Services

```
GET [api_version][org_href]/sec_policy/:pversion/virtual_services
```

Where :pversion represents the security policy version.

Query Parameters for the GET method

Use the following query parameters to restrict the results of the query.

Parameter	Description	Type	Required
name	Name on which to filter. This parameter supports partial matches.	String	yes
description	Filtering description. Supports partial matches.	String	no
external_data_set	The data source from which the resource originates. For example, if this virtual service information is stored in an external database.	String	no
external_data_reference	A unique identifier within the external data source. For example, if this virtual service information is stored in an external database.	String	no
max_results	Maximum number of results to return. Maximum limit for returned virtual services is 500. <div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;">  NOTE: If not specified, or a value greater than 500 is specified, the API returns a maximum of 500 results. </div>	Integer	no
service	Service URI	String	no
service_address.ip	IP address to assign to the virtual service.	String	no

Parameter	Description	Type	Required
service_address.fqdn	FQDN to assign to the virtual service.	String	no
service_ports.proto	Protocol to filter on.	String	no
service_ports_port	Specify a port or a port range to filter results. The range is from -1 to 65535.	Integer	no
usage	Include virtual service usage flags.	Boolean	no
labels	List of lists of label URIs encoded as a JSON string data type.	Array	no

Curl Command

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/sec_policy/active/virtual_services -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

Each individual virtual service returned is identified by the virtual service HREF. To GET, PUT, or DELETE an individual virtual service, identify the service by its HREF in the API call.

```
[
  {
    "href": "/orgs/7/sec_policy/draft/virtual_services/dc82e7e8-7304-4a25-9f49-97e11d7de5d0",
    "created_at": "2015-10-27T14:34:42.941Z",
    "updated_at": "2015-10-27T14:34:42.941Z",
    "deleted_at": null,
    "created_by": { "href": "/users/14" },
    "updated_by": { "href": "/users/14" },
    "deleted_by": null,
    "update_type": null,
    "name": "Tomcat",
    "description": null,
    "service": { "href": "/orgs/7/sec_policy/draft/services/101" },
    "labels": [ { "href": "/orgs/7/labels/82" } ],
    "ip_overrides": [],
    "apply_to": "host_only"
  },
]
```

```
{
  "href": "/orgs/7/sec_policy/draft/virtual_services/256525b6-e7c5-4ad7-b7af-
e70586aa1078",
  "created_at": "2016-06-23T00:33:58.995Z",
  "updated_at": "2016-06-23T00:33:58.995Z",
  "deleted_at": null,
  "created_by": { "href": "/users/54" },
  "updated_by": { "href": "/users/54" },
  "deleted_by": null,
  "update_type": null,
  "name": "Example",
  "description": null,
  "service": { "href": "/orgs/7/sec_policy/draft/services/107" },
  "labels": [],
  "ip_overrides": [],
  "apply_to": "internal_bridge_network"
},
{
  "href": "/orgs/7/sec_policy/draft/virtual_services/7b46fce0-4933-4e29-b86c-
7a2a71e686ed",
  "created_at": "2016-05-31T18:19:43.467Z",
  "updated_at": "2016-05-31T18:19:43.467Z",
  "deleted_at": null,
  "created_by": { "href": "/users/54" },
  "updated_by": { "href": "/users/54" },
  "deleted_by": null,
  "update_type": null,
  "name": "test",
  "description": null,
  "service": {"href": "/orgs/7/sec_policy/draft/services/218"},
  "labels": [
    { "href": "/orgs/7/labels/88" },
    { "href": "/orgs/7/labels/82" },
    { "href": "/orgs/7/labels/92" },
    { "href": "/orgs/7/labels/95" }
  ],
  "ip_overrides": [
    "192.0.1.3",
```

```

        "192.168.100.0/24"
    ],
    "apply_to": "host_only"
  },
  {
    "href": "/orgs/7/sec_policy/draft/virtual_services/1828d8ff-aeb7-4735-9975-
db692813d193",
    "created_at": "2017-10-29T19:41:15.648Z",
    "updated_at": "2017-10-29T19:41:15.648Z",
    "deleted_at": null,
    "created_by": {"href": "/users/14"},
    "updated_by": {"href": "/users/14"},
    "deleted_by": null,
    "update_type": null,
    "name": "Jawoo",
    "description": null,
    "service": { "href": "/orgs/7/sec_policy/draft/services/99" },
    "labels": [
      { "href": "/orgs/7/labels/88" },
      { "href": "/orgs/7/labels/82" },
      { "href": "/orgs/7/labels/92" },
      { "href": "/orgs/7/labels/101" }
    ],
    "ip_overrides": [
      "192.0.1.0",
      "192.168.100.0/24"
    ],
    "apply_to": "host_only"
  }
]

```

Get an Individual Virtual Service

Use this method to get an individual virtual service. In the call, you identify the virtual service by its HREF, which can be obtained when you get a collection of virtual services.

Use the following query parameters to restrict the results of the query:

Parameter	Description	Type	Required
org_id	Organization	Integer	Yes

Parameter	Description	Type	Required
pversion	Security policy version	String	Yes
virtual_service_id	ID of the virtual service	String	Yes
usage	Include virtual service usage flags	Boolean	No

URI to Get an Individual Virtual Service

```
GET [api_version][virtual_service_href]
```



NOTE:

For this method, you can get specify either draft or active for :pversion.

Curl Command

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/sec_policy/draft/virtual_services/89 -H "Accept: application/json" -u $KEY:$TOKEN
```

Response

```
{
  "href": "/orgs/2/sec_policy/draft/virtual_services/6005a35a-1598-4c7b-a827-be4390f46773",
  "created_at": "2017-12-11T20:56:28.629Z",
  "updated_at": "2017-12-11T21:07:10.407Z",
  "deleted_at": null,
  "created_by": { "href": "/users/9" },
  "updated_by": { "href": "/users/9" },
  "deleted_by": null,
  "update_type": "create",
  "name": "Docker1",
  "description": null,
  "service": { "href": "/orgs/2/sec_policy/draft/services/5" },
  "labels": [
    { "href": "/orgs/2/labels/18" },
    { "href": "/orgs/2/labels/26" },
    { "href": "/orgs/2/labels/126" }
  ],
  "ip_overrides": [
    "192.0.1.0",
  ]
}
```

```

        "192.168.100.0/24"
    ],
    "apply_to": "internal_bridge_network"
}
    
```

Create an Individual Virtual Service

Use this method to create an individual virtual service. Because a virtual service is a policy item, you must create it in the draft state, and then provision the change using the Security Policy API.

Once the virtual service is provisioned, you can use the service binding method to bind the virtual service to a workload.

URI to Create an Individual Virtual Service

```
POST [api_version][org_href]/sec_policy/draft/virtual_services
```

Request Properties

Parameter	Description	Type	Req
name	A name for the virtual service.	String	Yes
description	Optional description of the virtual service.	String	No
external_data_set	The data source from which the resource originates. For example, if this virtual service information is stored in an external database.	String	No
external_data_reference	A unique identifier within the external data source. For example, if this virtual service information is stored in an external database.	String	No
labels	Any labels you want to apply to the virtual service, identified by a label HREF.	Array	No
service	HREF of the service to bind.	Object	Yes
service_addresses	Configured under the property <code>service_address</code> . <ul style="list-style-type: none"> <code>ip</code>: IP address to assign to the virtual service <code>network</code>: Network URI for this IP address <code>port</code>: Port associated with the IP address for the service (1-65535) <code>fqdn</code>: FQDN to assign to the virtual service 	Object String String Integer String	No No No No

Parameter	Description	Type	Req
	Either <code>ip</code> or <code>port</code> can be specified, but not both. If <code>ip</code> is specified, either <code>network</code> or <code>port</code> must be specified; <code>port</code> cannot be used with <code>network</code> .		
<code>service_ports</code>	Service ports are: <ul style="list-style-type: none"> <code>port</code>: Port Number (integer 0-65535 or -1 for any port). Also, the starting port when specifying a range. <code>to_port</code>: High end of the port range inclusive if specifying a range. Do not send this parameter if you are not specifying a range. 	Object Integer Integer	No No No
<code>apply_to</code>	Allows you to choose if you want the rules associated with the virtual service to be enforced over a host or bridge network. <ul style="list-style-type: none"> <code>internal_bridge_network</code>: Virtual service rules are applied to a bridge network, interpreted in the FORWARD chain on Linux iptables (Windows platform is not supported in this release.) <code>host_only</code>: Virtual service rules are enforced on the host, interpreted by the INPUT/OUTPUT chains in Linux iptables. 	String	No
<code>ip_overrides</code>	Allows you to specify IP addresses or ranges (CIDR blocks) to use for programming the rules associated with the virtual service, instead of the IP address of the bound workload. This parameter is similar to <code>service_addresses</code> but the <code>ips</code> do not have associated ports or networks.	String	No

Request Body

To create a virtual service, you need the HREF of the service you want to “bind” to a workload. You can obtain a service HREF by calling a GET collection with the service

binding API.

Additionally, if you want to add labels to the virtual service, you need the HREF of each label you want to add. Label HREFs can be obtained by calling a GET collection with the Labels API. Labels are represented in the JSON request body as an array, opened and closed by square brackets ([]).

```
{
  "name": "MyVirtualService",
  "description": "Test",
  "service": { "href": "/orgs/7/sec_policy/draft/services/218" },
  "labels": [
    { "href": "/orgs/7/labels/88" },
    { "href": "/orgs/7/labels/82" },
    { "href": "/orgs/7/labels/92" },
    { "href": "/orgs/7/labels/95" }
  ]
}
```

Curl Command

To create a new virtual service:

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/virtual_services -H
"Content-Type: application/json" -u $KEY:$TOKEN.-d '{ "name": "MyVirtualService",
"description": "Test", "service": {"href": "/orgs/7/sec_
policy/draft/services/218"}, "labels": [{"href": "/orgs/7/labels/88"}, {"href":
"/orgs/7/labels/82"}, {"href": "/orgs/7/labels/92"}, {"href": "/orgs/7/labels/95"
}]}'
```

Create or Update Virtual Services Collection



NOTE:

Bulk operations are rate limited to 1,000 items per operation.

This method enables you to create a collection of virtual services in your organization using a single API call instead of creating individual services one at a time.

This capability is useful if you want to keep a set of PCE resources in sync with your internal representation of the resources, such as a configuration management data-base (CMDB) that holds the “source of truth” for your PCE resources.

After virtual services are created and the identifiers added to the service properties, you can get a collection of virtual services using query parameters that include the external data reference. You can also run an asynchronous query to get all virtual services through an offline job, which includes the external data references in the response.

The two properties you can use when creating virtual services, `external_data_set` and `external_data_reference` are UTF-8 strings with a maximum length of 255 characters each. The contents must form a unique composite key, meaning that both values of these properties are treated as a unique key. These two properties together are recognized as a unique key, even if one of them is left blank or set to zero.

URI to Create a Collection of Virtual Services

```
PUT [api_version][org_href]/sec_policy/draft/virtual_services/bulk_create
```

URI to Update a Collection of Virtual Services

```
PUT [api_version][org_href]/sec_policy/draft/virtual_services/bulk_update
```

Request Body

To create a collection of virtual services, pass a JSON object that describes the virtual service details. The request body and curl command for this method follow the same structure used to create an individual virtual service, only you add multiple virtual service JSON objects instead of just one.

Additionally, the `href` field must be present in the body for each virtual service that you are updating in the `bulk_update`.

Update a Individual Virtual Service

To update (PUT) an individual virtual service, you need to know the HREF of the virtual service you want to update. Virtual service HREFs are returned when you get a collection of virtual services.

URI to Update an Individual Virtual Service

```
PUT [api_version][org_href]/sec_policy/draft/virtual_services/virtual_service_id
```

Request Properties

The request properties for updating a virtual service are the same as those for [creating a virtual service](#).

Request Body

This example request body can be passed to update a virtual service to include a workload binding:

```
{
  "service": { "href": "/orgs/2/sec_policy/draft/services/91" },
  "labels": [
    { "href": "/orgs/2/labels/316" },
    { "href": "/orgs/2/labels/101" },
    { "href": "/orgs/2/labels/102" },
    { "href": "/orgs/2/labels/103" }
  ]
}
```

Curl Command

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/sec_
policy/draft/virtual_services/256525b6-e7c5-4ad7-b7af-e70586aa1078 -H "Content-
Type: application/json" -u $KEY:$TOKEN -d '
{"name":"test","description":null,"service":
{"href":"/orgs/2/labels/316"},"labels": [{"href":"/orgs/2/labels/101"},
{"href":"/orgs/2/labels/102"}, {"href":"/orgs/2/labels/103"}]}'
```

Service Bindings

After you create a virtual service and provision it, use the Service Binding API to bind the virtual service to a workload. When you apply your policy to a virtual service, the virtual service must be bound to a workload where that service is running. You can only specify one workload and one virtual service per service binding.

When you bind a virtual service to a workload with a service binding, you must specify the workload to which you want to bind the service. You can also optionally specify any port overrides if you want the virtual service to communicate over a different port than the default.

Unlike virtual services, the Service Binding API does not require provisioning to take effect.

**NOTE:**

Updating service bindings doesn't use a PUT method. To update it, delete it, and then POST a new service binding to replace it.

Service Binding API Methods

Functionality	HTTP	URI
Get a collection of service bindings	GET	[api_version][org_href]/service_bindings
Get an individual service binding	GET	[api_version][service_binding_href]
Create a service binding	POST	[api_version][org_href]/service_bindings
Delete an individual service binding	DELETE	[api_version][service_binding_href]

Create a Service Binding

This method creates one or more service bindings, which associate (or “bind”) a virtual service to a workload. When you call this method, you specify the virtual service and workload you want to bind, plus you can optionally specify port overrides to use a different port for the service.

The JSON request body for creating a service binding is an array, which allows you to create multiple service bindings with a single POST.

Make sure that before you create a service binding that the virtual service you want to bind to a workload has been published and is in the active policy state.

URI to Create a Service Binding

```
POST [api_version][org_href]/service_bindings
```

Request Parameters

The request body for creating a service binding is an array of service binding objects. Because this JSON request body is an array, you can create multiple service bindings in a single POST.

**NOTE:**

Make sure that the virtual service you are binding to a workload has been provisioned.

This is an example JSON representation of a single service binding:

```
[{"workload": {"href": "/orgs/1/workloads/45c69cf3-4cbb-4c96-81ee-70e94baea1b8"},
"virtual_service": {"href": "/orgs/1/sec_policy/draft/virtual_services/a735332e-5d31-4899-a3a5-fac7055e05c0"}, "port_overrides": [{"port": 14000, "protocol": 6, "new_port": 26000 }]}]
```

Curl Command

To create a single service binding:

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/service_bindings -H
"Content-Type:application/json" -u $KEY:$TOKEN -d '{"workload":
{"href":"/orgs/1/workloads/45c69cf3-4cbb-4c96-81ee-70e94baea1b8"}, "virtual_
service":{"href":"/orgs/1/sec_policy/draft/virtual_services/a735332e-5d31-4899-
a3a5-fac7055e05c0"}, "port_overrides":[{"port":14000,"protocol":6,"new_
port":26000}]}'
```

Request Body to Create Multiple Service Bindings

An example JSON request body for creating multiple service bindings with a different port number:

```
[{"workload": {"href": "/orgs/1/workloads/820efcdc-c906-46b9-9729-26bab7a53223"},
"virtual_service": {"href": "/orgs/1/sec_policy/draft/virtual_services/e38ce044-
d2ac-4d7f-aeec-16ef8fbd0b15"}, "port_overrides": [ {"port": 10000, "protocol": 6,
"new_port": 26000 } ]}, {"workload": {"href": "/orgs/1/workloads/820efcdc-c906-
46b9-9729-26bab7a53223"}, "virtual_service": {"href": "/orgs/1/sec_
policy/draft/virtual_services/e38ce044-d2ac-4d7f-aeec-16ef8fbd0b15"}, "port_
overrides": [ {"port": 11000, "protocol": 6, "new_port": 25000} ]}]
```

Service Binding Request Body

If you create more than one service binding with a single POST, all of the service bindings must be constructed properly or the POST will fail and no service bindings will be created.



NOTE:

The response of “failure” indicates the error, but it does not confirm that no service bindings have been created.

For example, if you use POST to create 10 service bindings, and one of the workloads referenced in the JSON payload uses an incorrect URI (HREF), the POST fails with an error message similar to the following message:

```
[ { "token": "invalid_uri", "message": "Invalid URI:
{/orgs/1/workloadzzz/820efcdc-c906-46b9-9729-26bab7a53223}" } ]
```

Get Individual or Collection of Service Bindings

You can use these methods to get one or more service bindings.

URI to Get a Collection of Service Bindings


```
GET [api_version][org_href]/service_bindings
```

URI to Get an Individual Service Binding

```
GET [api_version][service_binding_href]
```

Query Parameters

Parameter	Description	Type
virtual_service	The complete HREF of the virtual service referenced in the service binding.	String
workload	The complete HREF of the workload referenced in the service binding.	String
external_data_set	The data source from which the resource originates. For example, if this virtual service information is stored in an external database.	String
external_data_reference	A unique identifier within the external data source. For example, if this virtual service information is stored in an external database.	String
max_results	The maximum number of results you want to return when using the GET method. The maximum limit for returned service bindings is 500.	Integer

Parameter	Description	Type
	 <p>NOTE: If this parameter is not specified, or a value greater than 500 is specified, a maximum of 500 results are returned. To get more than 500 results, use an Asynchronous GET Collection.</p>	

Response Body

```
[
  {
    "href": "/orgs/7/service_bindings/287568ad-4a1f-4000-a9fb-e67d1dabce15",
    "virtual_service": {"href": "/orgs/7/sec_policy/active/virtual_
services/256525b6-e7c5-4ad7-b7af-e70586aa1078"},
    "workload": {"href": "/orgs/7/workloads/baef2547-2036-4e00-b6f7-
3f4be1f7669a",
    "name": null,
    "hostname": "AssetMgt-proc2",
    "deleted": false },
    "port_overrides": [{"new_port": 8080,"protocol": 6,"port": 3306}]
  },
  {
    "href": "/orgs/7/service_bindings/faebe7bf-0bb7-49a5-868e-
8297e038fa9e",
    "virtual_service": {"href": "/orgs/7/sec_policy/active/virtual_
services/7b46fce0-4933-4e29-b86c-7a2a71e686ed"},
    "workload": {"href": "/orgs/7/workloads/aee4381b-9836-45b6-b7ab-
aee246bf482f",
    "name": null,
    "hostname": "onlinestore-web2",
    "deleted": false },
    "port_overrides": []
  },
  {
    "href": "/orgs/7/service_bindings/924ad8c2-94bf-40f5-bc4c-
13474982bd00",
    "virtual_service": {"href": "/orgs/7/sec_policy/active/virtual_
services/256525b6-e7c5-4ad7-b7af-e70586aa1078"},

```

```
        "workload": {"href": "/orgs/7/workloads/69fd736b-cd21-4a4c-bdb9-132207c760ce",
                    "name": null,
                    "hostname": "test-us",
                    ": false },
                    "port_overrides": []
        }
    ]
```

Curl Command to Get an Individual Service Binding

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/service_bindings/xxxxxxxx-4a86-4dd4-b303-23f699d0ebbf -H "Accept: application/json" -u $KEY:$TOKEN
```

Curl Command to Get Service Binding Collection

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/service_bindings -H "Accept: application/json" -u $KEY:$TOKEN
```

Delete an Individual Service Binding

To delete both the service bindings and virtual services, delete the service bindings first, then delete the virtual services.

URI to Delete an Individual Service Binding

```
DELETE [api_version][service_binding_href]
```

Curl Command to Delete a Service Binding

Use this curl command to delete the service binding:

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/service_bindings/xxxxxxxx-4a86-4dd4-b303-23f699d0ebbf -u $KEY:$TOKEN
```

IP Lists

This Public Stable API can get, create, update, and delete IP lists.

IP lists can be used in rules to define sets of trusted IP address, IP address ranges, or CIDR blocks allowed into your datacenter that are allowed to access workloads in your network.

IP Lists API

Functionality	HTTP	URI
Get a collection of IP lists	GET	[api_version][org_href]/sec_policy/draft/ip_lists
Get an individual IP list	GET	[api_version][ip_list_href]
Create an IP list	POST	[api_version][org_href]/sec_policy/draft/ip_lists
Update an IP list	PUT	[api_version][ip_list_href]
Delete an IP list	DELETE	[api_version][ip_list_href]

Active vs Draft

This API operates on provisionable objects, which exist in either a draft (not provisioned) state or an active (provisioned) state.

Provisionable items include label groups, services, rulesets, IP lists, virtual services, fire-wall settings, and virtual servers. For these objects, the URL of the API call must include the element called `:pversion`, which can be set to either `draft` or `active`.

Depending on the method, the API follows these rules:

- For GET operations — `:pversion` can be `draft`, `active`, or the ID of the security policy.
- For POST, PUT, DELETE — `:pversion` can be `draft` (you cannot operate on active items) or the ID of the security policy.

Get IP Lists

This API allows you to get a collection of IP lists or a single IP list from an organization.

By default, the maximum number returned on a GET collection of IP lists is 500. If you want to get more than 500 IP lists, use an [Asynchronous GET Collection](#).

URI to Get Collection of IP Lists

```
GET [api_version][org_href]/sec_policy/draft/ip_lists
```

URI to Get an Individual IP List

```
GET [api_version][ip_list_href]
```

Query Parameters

Parameter	Description	Type
external_data_set	The data source from which the resource originates. For example, if this workload information is stored in an external database.	String
external_data_reference	A unique identifier within the external data source. For example, if this workload information is stored in an external database.	String
name	Name of the IP lists to return. Partial matches are supported.	String
description	Description used for the IP list. Partial matches are supported.	String
ip_address	IP address matching the IP lists to return. Partial matches are supported.	String
max_results	The maximum number of results you want to return when using the GET method. The maximum limit for returned IP lists is 500.	Integer

Curl Command to Get Collection of IP Lists

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/sec_policy/draft/ip_lists -H "Accept: application/json" -u $KEY:$TOKEN
```

Response Body

```
{
  href: "/orgs/2/sec_policy/draft/ip_lists/285"
  id: 285
  created_at: "2014-02-25T16:29:06Z"
  updated_at: "2014-02-25T16:29:07Z"
  deleted_at: null
  created_by: {
    href: "/users/76"
  }
  updated_by: {
    href: "/users/9"
  }
}
```

```
deleted_by: null
name: "Company Headquarters"
description: null
ip_ranges: [
  {
    description: ""
    from_ip: "209.37.96.18"
    to_ip: null
  }
]
}
{
  href: "/orgs/2/sec_policy/draft/ip_lists/306"
  id: 306
  created_at: "2014-04-09T18:37:21Z"
  updated_at: "2014-04-09T18:37:21Z"
  deleted_at: null
  created_by: {
    href: "/users/9"
  }
  updated_by: {
    href: "/users/9"
  }
  deleted_by: null
  name: "Dashboard Server"
  description: null
  ip_ranges: [
    {
      description: ""
      from_ip: "192.0.2.0"
      to_ip: null
    }
  ]
}
{
  href: "/orgs/2/sec_policy/draft/ip_lists/309"
  id: 309
  created_at: "2014-04-17T21:59:44Z"
```

```
updated_at: "2014-04-17T21:59:44Z"
deleted_at: null
created_by: {
  href: "/users/76"
}
updated_by: {
  href: "/users/76"
}
deleted_by: null
name: "Good IPs 2"
description: null
ip_ranges: [
  {
    description: "My good IPs for web app"
    from_ip: "192.0.2.0"
    to_ip: null
  }
]
```

Curl Command to Get an IP List

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/3/sec_policy/draft/ip_lists/312 -H "Accept: application/json" -u $KEY:$TOKEN
```

Create an IP List

This API allows you to create IP lists (allowlists) so they can be used for creating rules in rulesets. An IP list can contain a single IP address or an IP address range.



NOTE:

Denylist IP lists are not supported in this release.

URI to Create an IP List

```
POST [api_version][org_href]/sec_policy/draft/ip_lists
```

Request Properties

Example JSON request body for a single IP list:

```
{
  "name": "Good IPs",
  "ip_ranges": [
    {
      "description": "Good IPs allowed to access app server",
      "from_ip": "192.0.2.0"
    }
  ]
}
```

Curl Command to Create IP List

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/3/sec_policy/draft/ip_lists -H "Accept: application/json" -u $KEY:$TOKEN -d '{"name": "Good IPs", "ip_ranges":[{"description": "Good IPs allowed to access app server", "from_ip": "192.0.2.0"}]}'
```

Response Body

```
{
  href: "/orgs/2/sec_policy/draft/ip_lists/316"
  created_at: "2014-04-18T00:19:55Z"
  updated_at: "2014-04-18T00:19:55Z"
  deleted_at: null
  created_by: {
    href: "/users/11"
  }
  updated_by: {
    href: "/users/11"
  }
  deleted_by: null
  name: "Good IPs"
  description: null
  ip_ranges: [
    {
      description: "Good IPs"
      from_ip: "192.0.2.0"
      to_ip: null
    }
  ]
}
```



```
  ]  
}
```

Update an IP List

This API updates a specific IP list identified by its HREF. Get a collection of IP lists to find IP list HREFs .

Example IP list HREF:

```
/orgs/2/sec_policy/draft/ip_lists/316
```

URI to Update an IP List

```
PUT [api_version][ip_list_href]
```

Example Request Body to Update an IP List

```
{  
  "name": "Better IPs",  
  "list_type": "allow",  
  "ip_ranges": [  
    {  
      "description": "More allowed IPs for web app",  
      "from_ip" : "192.0.2.0"  
      "to_ip" : "24"  
    }  
  ]  
}
```

Curl Command to Update IP List

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/3/sec_policy/draft/ip_lists/312 -H "Content-Type: application/json" -u $KEY:$TOKEN -d '{ "name": "Better IPs", "list_type": "allow", "ip_ranges": [{"description": "Better IPs for web app", "from_ip": "192.0.2.0", "to_ip": "24"}]}'
```

Delete an IP List

This API removes an IP list from a organization:

URI to Delete an API List

```
DELETE [api_version][ip_list_href]
```

Curl Command to Delete IP List

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/sec_  
policy/draft/ip_lists/316 -u $KEY:$TOKEN
```

Virtual Servers and Load Balancers

Load Balancers

By applying labels to your load balancer's virtual servers, you can write rules that allows client workloads in front of the load balancer to communicate with the virtual IP address of the load balancer's virtual servers. By adding labels to the pool members behind a virtual sever, you can allow communication from the load balancer to the members of the pool. The source for this communication is determined by the load balancer. The Illumio Core programs policies on the load balancer to enforce security policy. The PCE uses the load balancer's REST APIs to read and write security policies to configure security rules.

The PCE supports configuration of two load balancer units if they are configured in Active/Standby or Active/Active modes. The PCE needs to be configured with the user name and password of an administrative user who has read-write access to all configurations on the load balancer.

The PCE configures new objects on the load balancer and does not alter any existing configurations. When a created object in the load balancer configuration is modified, the PCE detects it as tampering and modifies the configuration back to the intended state so that the correct security policy is enforced.

The Illumio Core dynamically adjusts policies on the load balancer based on application and topology changes in the datacenter so that the Illumio Core can enforce consistent security policy on load balancers across the datacenter and cloud environments, as well as show the application traffic in Illumination. The Illumio Core keeps track of the policy it programmed and reconfigures policy if it was altered on the load balancer manually or by other means.

The Illumio Core makes use of the following constructs on load balancers:

- LTM: iRules on LTM provide capability to restrict application access. When LTM is used as enforcement mechanism, the Illumio Core uses virtual-server based iRules and Datagroup Lists.
 - AFM: AFM provides stateful firewalling on BIG-IP. When AFM is used as an enforcement mechanism, the Illumio Core uses Network Firewall policies in the virtual server section and address-lists in the network firewall.
 - AVI: The Illumio Core uses the Network Security Policy rules to program AVI Vantage.



WARNING:
Configuring two F5 units in Active/Standby mode is supported

F5 BIG-IP Requirements

The Illumio Core uses its REST API to program F5 load balancers, which means that F5 needs to be running a software version that supports REST-API. The requirements include:

- BIG-IP 11.5.x or higher
- Utilize SNAT or Auto-map mode

AVI Vantage Requirements

- AVI Vantage 18.2.3 or higher

Virtual Servers

Virtual servers in the Illumio Core contain two parts:

- A virtual IP address (VIP) and port through which the service is exposed
- Local IP address(es) used to communicate with backend servers (pool members).

A virtual server is similar to a workload. It can be assigned labels and has IP addresses, but does not report traffic to the Illumio Core. Each virtual server has only one VIP. The local IP addresses are used as a source IP address for connections to the pool members (backend servers) when the virtual server is operating in SNAT mode or Auto mode. These IP addresses are likely to be shared by multiple virtual servers on the server load balancer.

A virtual server is identified by a set of labels. The consumers and providers for a virtual server can be assigned different labels, which could place them in the same group or a different group in Illumination.

Providers are allowed to have an incomplete label set (for example, only a Location label), so the providers can be in all groups within the specified location. As a result, a single virtual server can have providers in any group or in any number of groups in Illumination.

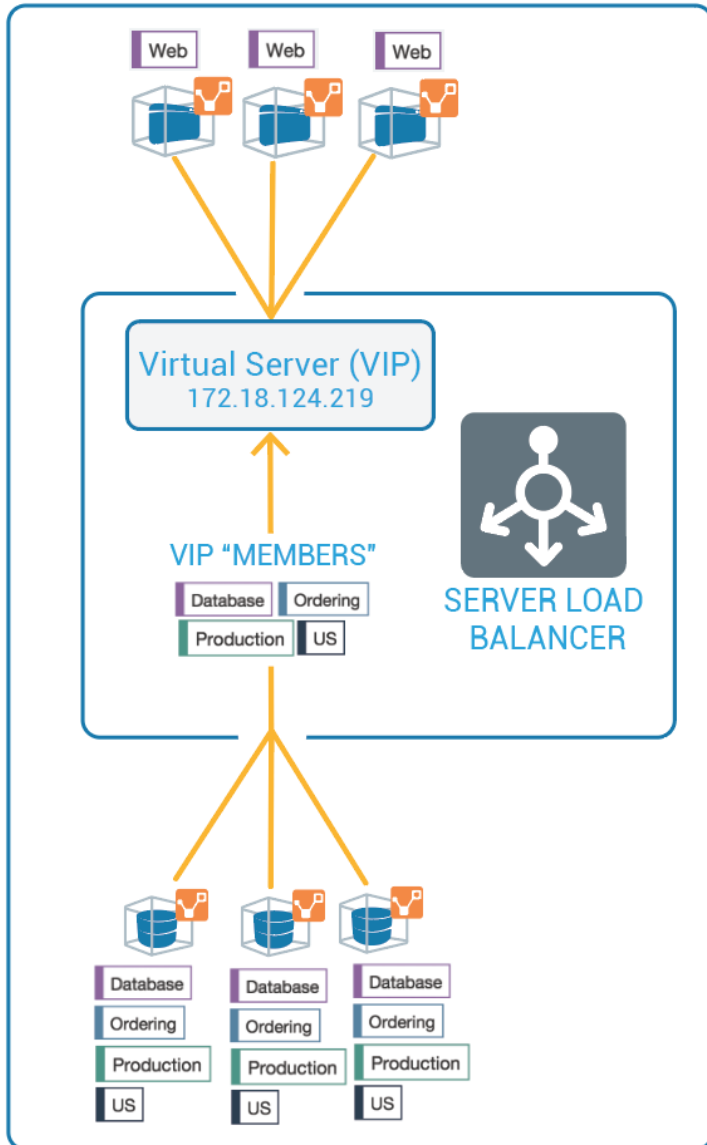
Virtual Server Members

The Illumio Core allows you to write rules to allow communication with workloads managed by a load balancer using labels.

When you configure load balancers in the PCE, it connects to the load balancer using the Illumio Core REST API. The PCE reads all the load balancer virtual servers configurations and populates the Discovered Virtual Servers tab of a load balancer's details page. Any virtual servers associated with the load balancer can be converted to a managed virtual server for use with the PCE. When you configure the virtual server in the PCE web console, you can apply labels to the virtual servers. After configuring a virtual server, you can write a rule that allows other clients to communicate with it.

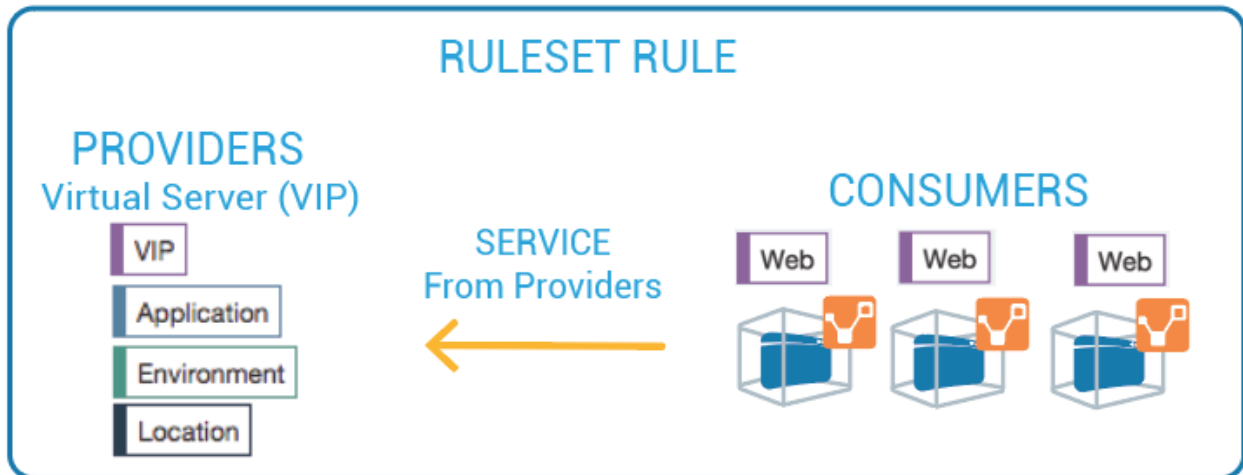
The members behind a virtual server are specified by configuring a set of labels in the virtual server configuration. A set of four Illumio labels can be applied on the Virtual Server Members tab, which is used to match the same set of labels on workloads in the virtual server's pool. If any of the workloads in the virtual server pool share the same set of four labels specified under the Virtual Server Members tab, then any rule you write with the virtual server also applies to the workload members.

In this diagram, you can see how the workloads that belong to the virtual server pool have the same labels specified on the Virtual Server Members tab:



Ruleset Rule for Virtual Server

This diagram illustrates the rule you can write after you label a virtual server and its members:



Configure Virtual Servers

After adding a load balancer to the PCE, you can manage its virtual servers. For each virtual server, you can add a complete set of the four Illumio labels: Role, Application, Environment, and Location. Adding labels to the virtual server enables you to add it in a rule.

You add the four Illumio labels to the Virtual Server's Members tab. When the labels specified under Virtual Server Members match labels of workloads in the virtual server pool, any rule you write with the virtual server are applied to the workload members.

Configuring a load balancer's virtual servers consists of these three settings:

- **Enforced or Not Enforced:** When you select Enforced, any rules you write using the labels associated with the virtual servers and any of its members are enacted. Selecting Not Enforced disables the labels and any policy written that affects the virtual server or its members is disabled.
- **Service:** Select the service to use for the rules that allow access to the virtual server. For example, HTTPD 80 TCP.
- **Labels:** You must apply one each of the four Illumio labels to the virtual server: Role, Application, Environment, and Location. Assigning labels enables the virtual server to be used in rules.

Virtual Server Limitations

- Illumination does not support location-level and application-level maps.
- If a single SNAT pool is shared between multiple virtual servers, the Illumination map does not render correctly.

- SNAT and Auto-map modes of F5 virtual servers are supported. Transparent mode is not supported.



NOTE: Before any virtual server configuration can go into effect, you need to provision your changes. See [Provisioning](#) for more information.

Virtual Server Methods and Parameters

Here are the methods to manage Virtual Servers.

Functionality	HTTP	URI
Get a list of Discovered Virtual Servers	GET	[api_version][org_href]/discovered_virtual_servers/:uuid
Get a list of Virtual Servers	GET	[api_version][org_href]/sec_policy/:version/virtual_servers
Get a specified Virtual Server	GET	[api_version][org_href]/sec_policy/:version/virtual_servers/:uuid
Create a Virtual Server object	POST	[api_version][org_href]/sec_policy/:version/virtual_servers
Modify the enforcement mode, labels, and backend/provider labels of a specified Virtual Server	PUT	[api_version][org_href]/sec_policy/:version/virtual_servers/:uuid

These are the parameters for Virtual Servers :

Parameter	Description	Type
name	The short friendly name of the virtual server	String
labels	Label URI	String
service	URI of associated service Service URI	String
providers	Includes Label and Workload information	String
mode	Management mode of the virtual server.	String
discovered_virtual_server	Corresponding discovered virtual server, server URI	String
created_at	The time (rfc3339 timestamp) at which this virtual server was created	date-time
updated_at	The time (rfc3339 timestamp) at which this virtual server was last updated	date-time
created_by	The URI of the user who created this virtual server	String
updated_by	The URI of the user who last updated this virtual	String

Parameter	Description	Type
	server	

Security Principals

Security principals are typically unique identifiers for Windows Advanced Directory groups, but they can also be unique identifiers for individuals. This Public Stable API allows you to get (one or many), create (one or bulk), update, and delete security principals.

An array of security principals HREFs can be passed into rules and rulesets in the `consuming_security_principals` array.

Security Principals API Methods

Security Principals Methods	HTTP	URI
Get Security Principals	GET	[api_version][org_href]/security_principals/
Get a Security Principal	GET	[api_version][org_href]/security_principals/sid
Create a Security Principal	POST	[api_version][org_href]/security_principals/
Bulk create Security Principals	PUT	[api_version][org_href]/security_principals/bulk_create
Update a Security Principal	PUT	[api_version][org_href]/security_principals/sid
Delete a Security Principal	DELETE	[api_version][org_href]/security_principals/sid

Query Parameters for GET

The only required parameter for this API method is `org_id`.

Use the following optional query parameters to restrict the results of the query.

Parameter	Description	Type	Required
name	Name of security principal to filter by	String	No
max_results	Maximum number of entries to return. Used only for Get Security Principals	Integer	No
sid	SID of the security principal to filter by	String	No

Get Security Principals

This GET command, by default, returns information for 100 security principals if `max_results` is not specified.

A maximum value of up to 500 can be specified for `max_results`. To return more than 500 security principals, see [Async Job Operations](#).

Curl Command to Get Security Principals

```
curl -X GET https://pce.my-company.com:8443/api/v2/security_principals -u $KEY:$TOKEN -H 'Accept: application/json'
```

Example JSON Response Body

```
{
  "sid": "string",
  "name": "string",
  "description": "string"
}
```

Get a Security Principal

This GET command returns information about one specific security principal indicated by its `sid`.

Curl Command to Get a Security Principal

```
curl -X GET https://pce.my-company.com:8443/api/v2/security_principals/{sid} -u $KEY:$TOKEN -H 'Accept: application/json'
```

Example JSON Response Body

```
{
  "sid": "string",
  "name": "string",
  "description": "string"
}
```

Query Parameters for POST

Use the following optional query parameters to create a Security Principal.

Parameter	Description	Type	Required
org_id	Organization	Integer	Yes
body	security_principals_get		No
sid	Active Directory SID (or any other unique identifier)	String	Yes
name	Name of security principal to filter by	String	Yes
description	A longer description of the security principal	String	No
sid	SID of the security principal to filter by	String	No

Create a Security Principal

This POST command on success returns the HREF of the created security principal.

Curl Command to Create a Security Principal

```
curl -X POST https://pce.my-company.com:8443/api/v2/security_principals -u $KEY:$TOKEN -H 'Content-Type: application/json'
```

Example JSON Request Body

```
{
  "sid": "string",
  "name": "string",
  "description": "string"
}
```

Query Parameters for PUT and DELETE

Use the following optional query parameters to restrict the results of the query.

Parameter	Description	Type	Required
org_id	Organization (all commands)	Integer	Yes
body	security_principals_get (bulk create and update)		No
sid	Security principal SID (update and delete)	String	Yes
name	Name of security principal (update only)	String	Yes
description	A longer description of the security principal (update only)	String	No

Bulk Create Security Principals

This PUT command creates multiple security principals.

A maximum of 2,000 security principals can be added in a call to this API. On success, this API returns an array containing the HREFs of the created security principals.

Curl Command to Bulk Create Security Principals

```
curl -X PUT https://pce.my-company.com:8443/api/v2/security_principals/bulk_create
-u $KEY:$TOKEN -H 'Content-Type: application/json'
```

Example JSON Request Body

```
[
  {
    "sid": "string",
    "name": "string",
    "description": "string"
  },
  {
    "sid": "string_2",
    "name": "string_2",
    "description": "string_2"
  }
]
```

Update a Security Principal

This PUT command updates a security principal.

Curl Command to Update a Security Principal

```
curl -X PUT https://pce.my-company.com:8443/api/v2/security_principals/{sid} -u
$KEY:$TOKEN -H 'Content-Type: application/json'
```

Example JSON Request Body

```
{
  "name": "string",
```

```
"description": "string"  
}
```

Delete a Security Principal

This command deletes a security principal.

Curl Command to Delete a Security Principal

```
curl -X DELETE https://pce.my-company.com:8443/api/v2/security_principals/{sid} -u  
$KEY:$TOKEN
```

This command returns 204 No Content for success.

Visualization

This chapter contains the following topics:

Vulnerabilities	285
Explorer	294
Bulk Traffic Loader	307

In addition to reviewing workloads and traffic with the PCE web console, you can analyze the traffic flows and get insight into the exposure to vulnerabilities using the Visualization API.

The Explorer API is used to search and analyze PCE traffic flows. It queries the PCE's traffic database and analyzes these flows for auditing, reporting, and troubleshooting. The VEN adds the DNS names to the flow summary logs and sends them to the PCE, while the Explorer API appends the DNS names to allow auditors and analysts to view them without performing reverse look-ups on random IP addresses.

Vulnerability Maps combine Illumio's Application Dependency Map with vulnerability data from Qualys Cloud Platform to provide insights into the exposure of vulnerabilities and attack paths across your applications.

Vulnerabilities

This Public Experimental API gets, creates, updates, and deletes vulnerabilities.

**NOTE:**

The Illumio Core Vulnerability Maps license is required to import Qualys report data into the Illumio PCE. For information about obtaining the Illumio Core Vulnerability Maps license, contact Illumio Support. When you obtain your license, you also receive information about how to install it.

Delete the License

To delete the vulnerability license, use the following CURL command from your CLI environment:

```
export API_KEY=api_key_username:api_key_secret
```

```
curl -i -H "Content-Type: application/json" https://pce_fqdn:8443/api/v2/orgs/org_id/licenses/9df01357-93cf-4f33-b720-e47bba783c55 -X DELETE -u $API_KEY
```

Replace the variables, which are entered in **blue bold**.

Vulnerabilities API Methods

Functionality	HTTP	URI
Get vulnerabilities	GET	[api_version][org_href]vulnerabilities
Get an individual vulnerability	GET	[api_version][org_href][vulnerabilities_href]
Create an individual vulnerability	PUT	[api_version][org_href][vulnerabilities_href]
Modify an individual vulnerability	PUT	[api_version][org_href][vulnerabilities_href]
Delete an individual vulnerability	DELETE	[api_version][org_href][vulnerabilities_href]

Get Collection of all Vulnerabilities

In this example, the maximum number of vulnerability reports is set to 2. Not using this query parameter in this GET method would return all the vulnerability reports up to a maximum of 500. For more than 500 vulnerabilities, use an [Asynchronous GET Collection](#).

Curl Command to Get Collection of Vulnerabilities

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/vulnerabilities -H
'Accept: application/json' -u $KEY:$TOKEN
```

Parameter	Description	Parameter Type	Data Type
max_results	The maximum number of vulnerabilities returned by a call to GET /vulnerabilities. (Optional. If not specified, all vulnerabilities are returned up to a maximum of 500.)	Query	Integer

Response Body

```
[
  {
    "href": "/orgs/2/vulnerabilities/qualys-xxxxxebe7e17",
    "name": "Host Scan Time",
    "score": 37,
    "description": "{\"severity\":\"1\"}",
    "cve_ids": [],
    "created_at": "2017-12-21T19:15:48.000Z",
    "updated_at": "2017-12-21T19:17:26.000Z",
    "created_by": null,
    "updated_by": null
  },
  .....
]
```

Get a Vulnerability

Curl Command to Get an Individual Vulnerability

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/vulnerabilities/qualys-xxxxxebe7e18 -H 'Accept:
application/json' -u $KEY:$TOKEN
```

Get an Individual Vulnerability

Parameter	Description	Parameter Type	Data Type
reference_id	The ID of the vulnerability to return by GET /vulnerabilities/{reference_id}.	Path	String

Response Body

```
{
  "href": "/orgs/2/vulnerabilities/qualys-xxxxxebe7e18",
  "name": "Host Scan Time",
  "score": 37,
  "description": "{\"severity\":\"1\"}",
  "cve_ids": [],
  "created_at": "2017-12-21T19:15:48.000Z",
  "updated_at": "2017-12-21T19:17:26.000Z",
  "created_by": null,
  "updated_by": null
}
```

Create or Update a Vulnerability

Curl Command to Create or Update Vulnerability

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/7/vulnerabilities/qualys-xxxxxebe7e18 -H 'Content-Type: application/json' -u $KEY:$TOKEN -d '{"score": 50, "cve_ids": ["CVE-2012-xxxx", "CVE-2017-xxxx"], "description": "My vulnerability test."}'
```

Parameter	Description	Parameter Type	Data Type
reference_id	The ID of the vulnerability. The reference_id is the last element of the href property returned by a call to GET /vulnerabilities.	Path	String
score	The normalized score of the vulnerability in the range of 0 to 100 inclusive. CVSS Score can be used here with a 10x multiplier.	Body	Integer

Parameter	Description	Parameter Type	Data Type
name	The title/name of the vulnerability.	Body	String
cve-ids	The cve_ids for the vulnerability.	Body	[String]
description	An arbitrary field to store details about the vulnerability class.	Body	String

Example Request Body

```
{
  "score": 50,
  "cve_ids": ["CVE-2012-xxxx", "CVE-2017-xxxx"],
  "description": "My vulnerability test."
}
```

Response

On success, the system displays HTTP/1.1 204 No Content.

Delete a Vulnerability

To delete an individual vulnerability, specify its HREF, which can be obtained from the response from GET /vulnerabilities.

Curl Command to Delete Vulnerability

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/7/vulnerabilities/qualys-xxxxxebe7e18 -u $KEY:$TOKEN
```

Request Parameter

Parameter	Description	Parameter Type	Data Type
reference_id	The reference ID of the vulnerability. The last element of the href property of a vulnerability returned by a call to GET /vulnerabilities.	Path	String

Vulnerability Reports

This Public Experimental API creates, updates, and deletes vulnerability reports.

**NOTE:**

An Illumio Core Vulnerability Maps license is required to import Qualys report data into the Illumio PCE. For information about obtaining the Illumio Core Vulnerability Maps license, contact Illumio Support. When you obtain your license, you also receive information about how to install it.

Vulnerability Reports API Methods

HTTP	Functionality	URI
GET	Get a collection of vulnerability reports	[api_version][org_href]/vulnerability_reports
GET	Get an individual vulnerability report	[api_version][vulnerability_reports_href]
POST	Create an individual vulnerability report	[api_version][vulnerability_reports_href]
PUT	Update an individual vulnerability report	[api_version][vulnerability_reports_href]
DELETE	Delete an individual vulnerability report	[api_version][vulnerability_reports_href]

Get a Collection of Vulnerability Reports

This method gets a collection of all vulnerability reports in your organization.

By default, the maximum number returned by a GET collection of vulnerability reports is 500. For more than 500 vulnerability reports, use an [Asynchronous GET Collection](#).

Curl Command to Get Collection of Vulnerability Reports

In this example, the maximum number of vulnerability reports is set to 2. Not using this query parameter in this GET method would return all the vulnerability reports up to a maximum of 500.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/vulnerability_reports
-H 'Accept: application/json' -u $KEY:$TOKEN
```

Query Parameter to Get a Collection of Vulnerability Reports

Parameter	Description	Parameter Type	Data Type
max_results	The maximum number of vulnerability reports returned by a call to GET /vulnerability_reports.	Query	Integer

Parameter	Description	Parameter Type	Data Type
	Optional. If not specified, by default, all vulnerability reports are returned up to a maximum of 500.		

Response Body

```
[
  {
    "href": "/orgs/2/vulnerability_reports/qualys-report-12345",
    "report_type": "qualys",
    "name": "my-report-2017-12-21-19-15-47",
    "created_at": "2017-12-21T19:15:48.000Z",
    "updated_at": "2017-12-21T19:15:48.000Z",
    "num_vulnerabilities": 4887,
    "created_by": null,
    "updated_by": null
  },
  {
    "href": "/orgs/2/vulnerability_reports/qualys-report-12346",
    "report_type": "qualys",
    "name": "my-report-2017-12-21-19-17-15",
    "created_at": "2017-12-21T19:17:15.000Z",
    "updated_at": "2017-12-21T19:17:15.000Z",
    "num_vulnerabilities": 1776,
    "created_by": null,
    "updated_by": null
  }
]
```

Get a Vulnerability Report

Curl Command to Get Vulnerability Report

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/vulnerability_reports/qualys-report-123456 -H 'Accept: application/json' -u $KEY:$TOKEN
```

Request Parameter to Get an Individual Vulnerability Report

The following required path parameter restricts the results of the GET command to the specified vulnerability report.

Parameter	Description	Parameter Type	Data Type
reference_id	The ID of the vulnerability report (this is the last element in the vulnerability report HREF returned by a call to GET /vulnerability_reports).	Path	String

Response Body

```
{
  "href": "/orgs/2/vulnerability_reports/qualys-report-123456",
  "report_type": "qualys",
  "name": "my-report-2017-12-21-19-17-15",
  "created_at": "2017-12-21T19:17:15.000Z",
  "updated_at": "2017-12-21T19:17:15.000Z",
  "num_vulnerabilities": 1776,
  "created_by": null,
  "updated_by": null
}
```

Create or Update a Vulnerability Report

Curl Command to Update a Vulnerability Report

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/7/vulnerability_reports/qualys-report-123456 -H 'Content-Type: application/json' -u $KEY:$TOKEN -d '{"name": "My vulnerability report", "report_type": "qualys"}'
```

Request Parameters

Parameter	Description	Parameter Type	Data Type
reference_id	The ID of the vulnerability report (this is the last element in the vulnerability report HREF returned by a call to GET /vulnerability_reports).	Path	String
name	User generated the name of the vul-	Body	Integer

Parameter	Description	Parameter Type	Data Type
	nerability report.		
report_type	A string representing the type of the report.	Body	String
authoritative	Boolean value specifies whether a report is authoritative or not.	Body	[String]
scanned_ips	The ips on which the scan was performed. Enforced 100K maxitem limit.	Body	String
detected_vulnerabilities	<p>An array of parameters, of which ip_address, workload, and vulnerability are required. Enforced 100K maxitem limit.</p> <p>ip_address: (Required) The IP address of the host where the vulnerability is found (string)</p> <p>port: The port associated with the vulnerability (integer)</p> <p>proto: The protocol that is associated with the vulnerability (integer)</p> <p>workload: (Required) The URI of the workload associated with this vulnerability (string)</p> <p>vulnerability: (Required) The URI of the vulnerability class associated with this vulnerability (string)</p>	Array	
external_data_reference	(PUT only) This parameter supports third-party reference data		
state	(PUT only) Enables deletion, addition, or updating of vulnerabilities		
exported_at	(PUT only) Saves the timestamp for the next delta pull.		

Example Request Body

```
{
  "name": "My vulnerability report",
```

```
"report_type": "qualys",
"authoritative": true
}
```

Response

On success, the system displays HTTP/1.1 204 No Content.

Delete a Vulnerability Report

To delete an individual vulnerability report, specify the last element of its HREF, which can be obtained from the response from GET `/vulnerabilities`.

Curl Command to Delete Vulnerability Report

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/7/vulnerability_
reports/qualys-report-2017-12-21-19-17-15 -u $KEY:$TOKEN
```

Request Parameter

Parameter	Description	Parameter Type	Data Type
reference_id	The ID of the vulnerability report (this is the last element in the vulnerability report HREF returned by a call to GET <code>/vulnerability_reports</code>).	Path	String

Explorer

The Public Experimental API searches and analyzes PCE traffic flows for auditing, reporting, and troubleshooting. You can search for traffic flows between workloads or hosts, labeled workloads, or IP addresses, and you can restrict the search by specific port numbers and protocols.

Explorer API Method

Functionality	HTTP	URI
Search the PCE traffic data database to discover traffic patterns and write policy.	POST	<code>[api_version][org_href]/traffic_flows/traffic_analysis_queries</code>

URI for Using Explorer to Search Traffic Flows


```
POST [api_version][org_href]/traffic_flows/traffic_analysis_queries
```


Request Properties



NOTE:

The following properties are required unless otherwise noted.

Property	Description	Type
start_date	Starting date for the query. If left empty, the default interpretation is “today,” which is “now” minus 24 hours.	Date-time string (YYYY-MM-DDTHH:MM:SS)
end_date	Ending date for the query. If left empty, the default interpretation is “today,” which is “now” plus 24 hours.	Data-time string (YYYY-MM-DDTHH:MM:SS)
sources	<p>Source labels, workloads, or IP addresses to include or exclude in the search.</p> <p>The response can contain up to 50 matching IP addresses.</p> <div data-bbox="425 1203 503 1278" data-label="Image">  </div> <p>NOTE: The response returns sources as consumers.</p> <p>Optional sub-properties:</p> <ul style="list-style-type: none"> include: Targets that can be included are workloads, labels, or IP addresses identified by their HREF and structured as an array of JSON objects. If this property is left empty, then include means consider “ALL” or “ANY” of the object type. exclude: Targets that can be excluded are workloads, labels, or IP addresses identified by their HREF and structured as a JSON array. <ul style="list-style-type: none"> When IP List is present in the consumer part of 	Array of JSON objects

Property	Description	Type
	<p>traffic query, traffic from workloads that belong to that IP List will not be returned by default. If the user wishes to see that traffic, they need to set <code>exclude_workloads_from_ip_list_query: false</code></p> <ul style="list-style-type: none"> ◦ When IP List is present in the provider part of traffic query, traffic to workloads that belong to that IP List will not be returned by default. If the user wishes to see that traffic, they need to set <code>exclude_workloads_from_ip_list_query: false</code> 	
destinations	<p>Target labels, workloads, or IP addresses to include or exclude in the search.</p> <div data-bbox="407 800 1187 911" style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;">  NOTE: The response returns targets as providers. </div> <p>Required sub-properties:</p> <ul style="list-style-type: none"> • <code>include</code>: Targets that can be <i>included</i> are workloads, labels, or IP addresses identified by their HREF and structured as an array of JSON objects. If this property is left empty, then <code>include</code> means consider “ALL” or “ANY” of the object type. • <code>exclude</code>: Targets that can be <i>excluded</i> are workloads, labels, or IP addresses identified by their HREF and structured as a JSON array. If this property is left empty, then <code>exclude</code> means exclude “NONE” of the object types. 	Array of JSON objects
services	<p>Services (5-tuple of port/to_port/-proto/process/service) to include or exclude.</p> <p>Required properties:</p> <ul style="list-style-type: none"> • <code>include</code>: Destination ports that must be matched in the traffic flows to be returned. An empty value is interpreted as “match any or all 	

Property	Description	Type
	<p>port values.”</p> <ul style="list-style-type: none"> • <code>exclude</code>: Traffic flows with ports in this set are excluded from the final result. An empty value is interpreted as “exclude no ports” from the search. <p>Additional optional properties:</p> <ul style="list-style-type: none"> • <code>port</code>: Port Number (integer 0-65535). Also the starting port when specifying a range. • <code>to_port</code>: High end of port range inclusive if specifying a range. If not specifying a range then don't send this • <code>proto</code>: Protocol number • <code>process_name</code> • <code>windows_services_name</code> 	
<code>policy_decisions</code>	<p>Allows you to filter the query based on policy decision:</p> <ul style="list-style-type: none"> • <code>allowed</code>: Allowed traffic. • <code>potentially_blocked</code>: Allowed but potentially blocked traffic. This type of traffic occurs when a workload VEN is in the test policy state. • <code>blocked</code>: Blocked traffic. • <code>unknown</code> 	Array of strings
<code>max_results</code>	[Optional] Maximum number of flows to return	Integer
<code>exclude_workloads_from_ip_list_query</code>	Exclude workload traffic when IP List is provided either in consumer or provider part of traffic query	Boolean Default is: true

Response Properties

Property	Description	Required/Type
<code>src</code>	<p>Workloads (<code>workload</code>), IP addresses (<code>ip</code>), or labels (<code>label</code>) that are consuming the service of the traffic flow returned in the response.</p> <p><code>workload</code></p> <p><code>ip</code></p>	Required String String JSON list of strings

Property	Description	Required/Type
	label	
dst	Workloads (workload), IP addresses (ip), or labels (label) that are providing the service of the traffic connection in the response. workload ip label	Required String String JSON list of strings
service	Port, protocol, process, service name and user_name of this flow Additional required properties: <ul style="list-style-type: none"> port: Destination port proto :IANA protocol number process_name: Process Name for this flow windows_service_name: Windows Service Name for this flow user_name: User Name for this flow 	Required, Object Integer Integer String String String
num_connections	Number of traffic flows reported in the connection.	Required, Integer
policy_decisions	If there is a rule in place for the returned traffic data, this property indicates the policy decision for the flow. Indicates if the traffic flow was allowed, potentially blocked (but allowed), or blocked. <ul style="list-style-type: none"> allowed: Allowed traffic potentially_blocked: Allowed but potentially blocked traffic blocked: Blocked traffic unknown 	Required, String
flow_direction	Flow direction is inbound or outbound.	Required, String
transmission	Transmission type is broadcast or multicast.	Not required,

Property	Description	Required/Type
		String
timestamp_range	The time range during which the flows were reported in date-time format. Includes both of these sub-properties: <ul style="list-style-type: none"> first_detected: The first time this flow was detected within the time range specified by the query last_detected: The last time this flow was detected within the time range specified by the query 	Required, String
state	State of the flow, which can be one of the following values: <ul style="list-style-type: none"> active closed timed-out snapshot new incomplete 	Not required, String
dst_bo	Bytes sent till now by the destination over the flow during the interval	Integer
dst_bi	Bytes received till now by the destination over the flow during the interval	Integer
icmp_type	ICMP type for the flow	Integer
icmp_code	ICMP code for the flow	Integer

Example Explorer Search

One preliminary method of creating policy is to make sure that different datacenter environments are segmented from each other, such as separating development and testing environments from production environments. Before writing policy rules to allow or block this traffic, determine if there are any traffic flows between them.

With Explorer, you can ask the PCE:

Were there any traffic flows between my development and production environments, over any port besides port 80, excluding any Workloads that have a "Domain Controller" role label?

To construct this query, specify the following information:

- **sources:** In the JSON request, include the HREF of the "Development" environment label and exclude the HREF of the "Domain Controller" role label.
- **targets:** In the JSON request, include the HREF of the "Production" environment label and exclude any workloads that do not have a role label assigned to them.
- **ports:** Search for traffic over any port except port 80.

Request Body

```
    {
  "sources": {
    "include": [
      [
        {
          "label": {
            "href": "/orgs/45/labels/7",
            /* HREF of an environment label named Development. */
          }
        }
      ]
    ],
    "exclude": [
      {
        "label": {
          "href": "/orgs/45/labels/45",
          /* HREF of a role label named Domain Controller. */
        }
      }
    ]
  },
  "destinations": {
    "include": [
      [
        {
          "label": {
```

```
        "href": "/orgs/45/labels/8",
        /* HREF of an environment label named Production. */
    }
}
],
],
"exclude": [
    {
        "label": {
            "href": "/orgs/45/labels?key=role&exists=false"
            /* Exclude any workloads that have no role labels defined. */
        }
    }
]
},
"services": {
    "include": [
        [
            {
                "port": "21"
                /* Include port number 21 for TCP traffic. */
            },
            {
                "proto": "6"
                /* Include TCP number. */
            }
        ]
    ],
    "exclude": [
        [
            {
                "port": "5"
                /* Exclude port number 5 for UDP traffic. */
            },
            {
                "proto": "17"
                /* Exclude UDP protocol identified by number. */
            }
        ]
    ]
}
```

```
    ]
  ]
},
  "policy_decision": [
    {
      "allowed", "blocked"
    },
  ]
},
}
```

Curl Command to Search Traffic Data with Explorer

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/7/traffic_
flows/traffic_analysis_queries -H "Content-Type: application/json" -u $KEY:$TOKEN
-T explorer-example.json
```

The above command is listing contents of `explorer-example.json` to specify the query.

The example should be similar to the following:

explorer-example.json

```
    {
  "sources": {
    "include": [
      [
        {
          "label": {
            "href": "/orgs/1/labels/20"
          }
        }
      ]
    ],
    "exclude": [
      {
        "label": {
          "href": "/orgs/1/labels/21"
        }
      }
    ]
  }
}
```

```
    }
  ]
},
"destinations": {
  "include": [
    [
      {
        "label": {
          "href": "/orgs/1/labels/202"
        }
      }
    ]
  ],
  "exclude": [
    {
      "label": {
        "href": "/orgs/1/labels/205"
      }
    }
  ]
},
"services": {
  "include": [
    {
      "port": 80
    }
  ],
  "exclude": [
    {
      "port": 888
    }
  ]
},
"policy_decisions": ["allowed"],
"max_results": 2
}
```

Response

The response shows that there is a single traffic flow between your Development and Production Environment, the consumer of which is a workload identified by its IP address and labels, communicating with an unidentified IP address on TCP (number 6) over port 80.

```
{
  "src": {
    "ip": "10.2.3.161",
    "workload": {
      "href": "/orgs/1/workloads/3a5f2482-1188-4449-b035-0c7880486a4f",
      "hostname": "jorge-test-client5",
      "name": null,
      "os_type": linux,
      "labels": [{
        "href": "/orgs/1/labels/144",
        "key": "app",
        "value": "test_app_1"
      }, {
        "href": "/orgs/1/labels/148",
        "key": "env",
        "value": "test_env_1"
      }, {
        "href": "/orgs/1/labels/149",
        "key": "loc",
        "value": "test_place_1"
      }, {
        "href": "/orgs/1/labels/143",
        "key": "role",
        "value": "test_access_1"
      }
    ]
  },
  "dst": {
    "ip": "10.2.3.165",
    "workload": {
      "href": "/orgs/1/workloads/87a10385-7466-488f-bce4-899de08974a0",
      "hostname": "jorge-test-server-4",
      "name": null,

```



```
        "os_type": linux,
        "labels": [{
            "href": "/orgs/1/labels/145",
            "key": "app",
            "value": "test_app_2"
        }, {
            "href": "/orgs/1/labels/151",
            "key": "env",
            "value": "test_env_2"
        }, {
            "href": "/orgs/1/labels/149",
            "key": "loc",
            "value": "test_place_1"
        }, {
            "href": "/orgs/1/labels/143",
            "key": "role",
            "value": "test_access_1"
        }
    ]
    },
    "port": 14000,
    "proto": 17,
    "num_connections": 2,
    "policy_decision": "allowed",
    "timestamp_range": {
        "last_detected": "2020-03-03T00:38:12Z",
        "first_detected": "2020-03-03T00:38:12Z"
    }
}, {
    "src": {
        "ip": "10.2.3.161",
        "workload": {
            "href": "/orgs/1/workloads/3a5f2482-1188-4449-b035-0c7880486a4f",
            "hostname": "jorge-test-client5",
            "name": null,
            "os_type": linux,
            "labels": [{
                "href": "/orgs/1/labels/144",
```

```
        "key": "app",
        "value": "test_app_1"
    }, {
        "href": "/orgs/1/labels/148",
        "key": "env",
        "value": "test_env_1"
    }, {
        "href": "/orgs/1/labels/149",
        "key": "loc",
        "value": "test_place_1"
    }, {
        "href": "/orgs/1/labels/143",
        "key": "role",
        "value": "test_access_1"
    }
  ]
}
},
"dst": {
  "ip": "10.2.3.165",
  "workload": {
    "href": "/orgs/1/workloads/87a10385-7466-488f-bce4-899de08974a0",
    "hostname": "jorge-test-server-4",
    "name": null,
    "os_type": linux,
    "labels": [{
      "href": "/orgs/1/labels/145",
      "key": "app",
      "value": "test_app_2"
    }, {
      "href": "/orgs/1/labels/151",
      "key": "env",
      "value": "test_env_2"
    }, {
      "href": "/orgs/1/labels/149",
      "key": "loc",
      "value": "test_place_1"
    }, {
      "href": "/orgs/1/labels/143",
```

```

        "key": "role",
        "value": "test_access_1"
      }}
    }
  },
  "port": 14000,
  "proto": 6,
  "num_connections": 6,
  "policy_decision": "allowed",
  "timestamp_range": {
    "last_detected": "2020-03-03T00:38:10Z",
    "first_detected": "2020-03-03T00:38:10Z"
  }
}

```

Bulk Traffic Loader

This Public Experimental API allows you to upload a snapshot of static traffic connections from a non-Illumio environment to the PCE so you can visualize what your network traffic data looks like in the PCE Illumination map without having to install VENs on your workloads.



NOTE:

Bulk operations are rate limited to 1,000 items per operation.

Bulk Traffic Loader API Methods

Functionality	HTTP	URI
Upload a collection of traffic data to the PCE	POST	[api_version][org_href]/agents/bulk_traffic_flows

Workflow to Upload Bulk Traffic

To upload bulk traffic data to the PCE:

1. Obtain network traffic information from your environment (using netstat or another network utility). For each traffic connection, obtain the source IP address, destination IP address, destination port, and protocol.
2. Create unmanaged workloads in the PCE.

3. In the PCE, map both the source IP address and destination IP address for each traffic flow to the unmanaged workloads. An unmanaged workload is required for each source and destination IP address that represents traffic in the Illumination map. This must be done before you load the traffic data.
4. Use the Bulk Traffic Loader API to upload the CSV-formatted traffic data into the PCE.

Create Collection of Unmanaged Workloads

Using this method, you can create a collection of unmanaged workloads in your organization using a single API call, rather than having to individually create individual workloads one at a time.

When you create a collection of workloads using this method, these workloads are recorded in the PCE as “unmanaged,” which means that no VEN has been installed on the workloads and no policy can be applied to them.

Create an unmanaged workload for each source IP address and destination IP address before you use the Bulk Traffic Loader API.

URI for Creating a Collection of Unmanaged Workloads

```
PUT [api_version][org_href]/workloads/bulk_create
```


Request Body

When you create a collection of workloads, you pass a JSON request body that defines the workload details.

Although this example illustrates the request body for a single workload, you can add as many workloads as you want using the `bulk_create` method.

Request Properties

Property	Description	Type	Required
<code>name</code>	Short, friendly name of the workload.	String	Yes
<code>description</code>	Long description of the workload.	String	No
<code>external_data_set</code>	An external data set identifier. For example, if this workload information is stored in an external database.	String	No
<code>external_data_reference</code>	An external data set reference path. For example, if this workload information is stored in an external database.	String	No

Property	Description	Type	Required
hostname	The hostname reported by the workload.	String	Yes
service_principal_name	The Kerberos Service Principal Name (SPN). This property is only relevant if you have configured Kerberos in your environment to authenticate VEN-to-PCE communication.	String	No
distinguished_name	The X.509 Subject distinguished name, used if you want this workload to use machine authentication between VENS and other hosts. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  NOTE: This property has an API exposure level of Public Experimental, which means it is not intended for production use and might change in future releases. For more information, see API Classification and Version. </div>	String	No
public_ip	The public IP address of the workload.	String	No
interfaces	List of all functioning network interfaces on the workload. Properties for workload network interfaces: <ul style="list-style-type: none"> • name: The interface name with a string data type • link state: The state of the interface connection with a string data type; one of three values: <ul style="list-style-type: none"> ◦ up: Interface is communicating. ◦ down: Interface is not communicating. ◦ unknown: State of the interface is unknown. • address: The IP address assigned to the interface with a string data type 	Array	No

Property	Description	Type	Required
	<ul style="list-style-type: none"> <code>cidr_block</code>: The number of bits in the subnet (for example, /24 is 255.255.255.0) with an integer data type <code>default_gateway_address</code>: The default IP address of the default gateway, either IPv4 or IPv6 with an integer data type <code>friendly_name</code>: The friendly name given to the interface with a string data type 		
<code>service_provider</code>	Name of the service provider that is hosting the workload.	String	No
<code>data_center</code>	The name of the data center where the workload is being hosted.		
<code>data_center_zone</code>	The zone inside the data center hosting the workload.	String	No
<code>os_id</code>	Unique OS identifier given by Illumio to the workload.	String	No
<code>os_detail</code>	Additional descriptive information about the workload OS	String	No
<code>online</code>	Indicates whether the workload is online and can communicate with the PCE.	Boolean.	No
<code>labels</code>	Labels that are attached to the workload.	Array.	No
"agent" (VEN) properties hash:			
<code>mode</code>	<p>This property defines the policy state of the workload's VEN. There are three possible states:</p> <p><code>idle</code>: A state in which the VEN does not take control of the workload's iptables (Linux) or Windows firewall (Windows), but rather provides the PCE relevant details about the workload, such as the workload IP address, operating system, and geographical location.</p> <p><code>illuminated</code>: The Illuminated policy state has two variations: <i>Build</i> and <i>Test</i> (configurable in the PCE web console):</p>	String	No

Property	Description	Type	Required
	<ul style="list-style-type: none"> <i>Build</i> is a state in which the VEN introspects all open ports on a workload and reports the flow of traffic between it and other workloads to the PCE. In this state, the PCE displays the flow of traffic to and from the workload. The PCE provides insight into the data center and the applications running in it. No traffic is blocked in this state. <i>Test</i> is a state which allows you to apply all rules and visualize all the traffic that would be blocked if you placed the workloads into the enforced policy state. No traffic is blocked in this state. <p>enforced: A state in which all rules definitions and restrictions are enforced.</p>		
log_traffic (deprecated)	If set to True, then traffic flows from the VEN are logged.	Boolean	No
visibility_level	<p>The amount of data the VEN collects and reports to the PCE from a workload in the enforced mode (policy state), so you can control resource demands on workloads.</p> <p>The higher levels of detail are useful for visualizing traffic flows in the Illumination map inside the PCE web console. If this parameter is not set, then VEN visibility level is set to <code>flow_summary</code>.</p> <ul style="list-style-type: none"> <code>flow_summary</code>: (“High Detail” in the PCE web console) The VEN collects traffic connection details (source IP, destination IP, protocol, and source and destination port) for both allowed and blocked connections. This option creates traffic links in the Illumination map and is typically used during the building and testing phase of your security 		

Property	Description	Type	Required
	<p>policy.</p> <ul style="list-style-type: none">• <code>flow_drops</code>: (“Less Detail” in the PCE web console.) The VEN only collects traffic connection details (source IP, destination IP, protocol, and source and destination port) for blocked connections. This option provides less detail for Illumination but demands fewer system resources from a workload and is typically used for policy enforcement.• <code>flow_off</code>: (“No Detail” in the PCE web console.) The VEN does not collect any details about traffic connections. This option provides no Illumination detail and demands the least amount of resources from workloads. This mode is useful when you are satisfied with the rules that have been created and do not need additional overhead from observing workload communication.		

JSON Payload

```
{
  "name": "workload 0",
  "description": "workload desc 0",
  "service_principal_name": "spn 0",
  "hostname": "workload-0.example.com",
  "public_ip": "24.1.1.1",
  "ephemeral": false,
  "external_data_set": "cldb",
  "external_data_reference": "0",
  "interfaces": [
    {
      "name": "eth0",
      "link_state": "up",
      "address": "10.0.0.2",
      "cidr_block": 24,
```



```
    "ip_version": 4,
    "default_gateway_address": "10.0.0.1",
    "friendly_name": "wan"
  }
],
"labels": [
  {"href": "/orgs/2/labels/1"},
  {"href": "/orgs/2/labels/9"}
],
"service_provider": "service provider",
"data_center": "central data center",
"os_id": "os id 0",
"os_detail": "os detail 0",
"online": true,
"agent": {
  "config": {
    "enforcement_mode": "full",
    "visibility_level": "flow_summary"
  }
}
}
```

Curl Command to Create Unmanaged Workload Collection

This curl example illustrates how to create two workloads using the Workload Bulk Create API. This example assumes you have already created API keys for PCE authentication.

```
curl -i -X PUT https://pce.mycompany.com:8443/api/v2/orgs/2/workloads/bulk_create
-H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"name":"web_
app1","description": "workload desc 0","service_principal_name":"spn
0","hostname":"workload-0.example.com", "public_
ip":"24.1.0.1","ephemeral":false,"external_data_set":"cldb","external_data_
reference": "0","interfaces":[{"name":"eth0","link_
state":"up","address":"10.0.0.2","cidr_block":24, "ip_version":4,"default_gateway_
address":"10.0.0.1","friendly_name":"wan"}],"labels":[{"href":"/orgs/2/labels/1"},
{"href":"/orgs/2/labels/9"}],"service_provider":"service provider","data_
center":"central data center","os_id":"os id 0","os_detail": "os detail
```

```
0", "online": true, "agent": {"config": {"enforcementmode": "full", "visibility_
level": "flow_summary"}}}]'
```

Upload Bulk Traffic Flows



NOTE:

Illumio recommends limiting bulk traffic operations to a maximum of 1,000 traffic flows per operation.

This method uses a POST command and requires a payload of multiple network traffic connections (source IP address, destination IP address, destination port, and protocol for each flow) passed as a single string of data in the CSV format.

URI for Uploading Bulk Traffic Flows

```
POST [api_version][org_href]/agents/bulk_traffic_flows
```

Header

When making this call, specify in the header that the data type is CSV and provide the CSV file format version. For example:

```
X-Bulk-Traffic-Load-CSV-Version:1
```

CSV Input Format

The traffic flow data that this method sent to the PCE in the CSV format must contain the following information:

Column	Type	Comment
src	String (IP address)	Must be a correctly-formatted IPv4 or IPv6 address. This IP address must map to an unmanaged workload in the PCE before you load the traffic data to the PCE.
dst	String (IP address)	Must be a correctly-formatted IPv4 or IPv6 address. This IP address must map to an unmanaged workload in the PCE before you load the traffic data to the PCE.

Column	Type	Comment
destination port	Integer	Destination port (port exposed by the destination IP address).
protocol	Integer	Protocol integer that must be converted to its Assigned Internet Protocol Numbers (for example., "tcp," must be converted it to its IANA number 6) before you make the API call.

**NOTE:**

The network protocol for each connection must be translated to its assigned internet protocol number before you use this API to send the data. These protocol numbers can be found on the Internet Assigned Numbers Authority (IANA) website.

CSV Payload

For each network traffic connection, provide the source IP address, destination IP address, destination port, and network protocol (translated to its assigned internet protocol number). Each new connection entry must be specified by an `\n` to indicate a line break in the CSV file.

Example of a `flows.csv` file:

```
"192.3.106.142,192.3.66.238,10123,6\n192.3.106.142,192.5.179.150,5723,6\n192.3.199.114,192.4.114.14,49241,6 \
192.4.113.239,192.4.113.240,1573,6\n192.4.205.21,192.1.189.138,13724,6\n192.4.205.21,192.1.189.138,1556,6 \
192.1.189.137,192.1.189.136,1563,6\n192.1.189.137,192.1.189.136,1563,6\n192.1.189.137,192.1.189.135,6200,6 \
192.1.189.137,192.1.189.136,1575,6\n"
```

Curl Commands to Upload Bulk Traffic Flows from CSV

If you have a pre-formatted CSV file, you can use the following curl command to upload the traffic flow data (where `@/path/flows.csv` represents the network path of the `.csv` file):

```
curl -i -X POST https://pce.illum.io:8443/api/v2/orgs/2/agents/bulk_traffic_flows
-H "X-Bulk-Traffic-Load-CSV-Version: 1" -u $KEY:$TOKEN --data-binary "@/
{path}/flows.csv"
```

This curl example shows two traffic connections being uploaded to the PCE:

```
curl -i -X POST https://mypce.mycompany.com:8443/api/v2/orgs/2/agents/bulk_
traffic_flows -u $KEY:$TOKEN -H "X-Bulk-Traffic-Load-CSV-Version: 1" --data
"10.0.0.1,10.0.0.2,23,6\n10.0.0.4,10.0.0.5,45,6"
```

This curl example shows a CSV file with traffic connections uploaded to the PCE:

```
curl -i -X POST https://devtest167.ilabs.io:8443/api/v2/orgs/1/agents/bulk_
traffic_flows -T "bulk_upload.csv" -H "X-Bulk-Traffic-Load-CSV-Version: 1" -H
"Content-Type: text/csv" -u $KEY:$TOKEN
```

Response Body

After successfully uploading traffic data with this API, an HTTP Status code of “201 Created” is returned along with a JSON object in the response body that contains a summary of the operation:

- `num_flows_received`: Total number of flows sent to the PCE
- `num_flows_failed`: Number of flows that failed to be uploaded
- `failed_flows`: Array of CSV lines that failed to parse or match a Workload

Example:

```
{
  "num_flows_received": 10,
  "num_flows_failed": 0
  "failed_flows": []
}
```

Chapter 10

Workloads

This chapter contains the following topics:

Pairing Profiles and Pairing Keys	318
Workload Operations	327
Workload Interfaces	340
Workload Settings	344
Workload Bulk Operations	347
VEN Operations	353
Agents on Workloads	358
Blocked Traffic to and from Workloads	362
Filtering and Aggregating Traffic	362

The Workloads APIs allow you to get information about workloads and network interfaces and to identify unauthorized traffic to or from workloads. Use the Workloads APIs to perform workload-related operations, such as pair workloads, configure pairing profiles, and obtain pairing keys.

Configure pairing profiles to apply properties to workloads as they pair with the PCE, such as what labels to apply. By configuring a pairing profile, you obtain a unique pairing key that identifies the VEN. Pair workloads to install VENs on them. The VEN reports detailed workload information to the PCE, such as which services are running on the workload.

Pairing Profiles and Pairing Keys

The Public Experiment API for pairing profiles gets, creates, updates, and deletes pairing profiles.

The Public Stable API for pairing keys creates a pairing key to use for pairing workloads.

About Pairing Profiles and Keys

Pairing profiles apply specific properties to workloads as they pair with the PCE, such as labels and the workload policy state.

When you configure a pairing profile, the pairing script contains a unique pairing key at the end of the script (activation-code) that identifies the VEN securely so it can authenticate with the PCE. You can configure a pairing key for one-time use or more, and you can also set time and use limits.

The Pairing Key API can generate a new pairing key from a specified pairing profile.

Pairing Profile Methods

Functionality	HTTP	URI
Get a collection of pairing profiles	GET	[api_version][org_href]/pairing_profiles
Create an individual pairing profile	POST	[api_version][org_href]/pairing_profiles
Update an individual pairing profile	PUT	[api_version][pairing_profile_href]
Delete an individual pairing profile	DELETE	[api_version][pairing_profile_href]

Get Pairing Profiles

This method allows you to get a collection of all pairing profiles in your organization or just an individual pairing profile.

By default, the maximum number returned on a GET collection of pairing profiles is 500. For more than 500 pairing profiles, use an [Asynchronous GET Collection](#).

URI to Get a Collection of Pairing Profiles

```
GET [api_version][org_href]/pairing_profiles
```

Curl Command to Get Collection of Pairing Profiles

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/1/pairing_profiles -H
'Accept: application/json' -u $KEY:'TOKEN'
```

Query Parameters

The following optional query parameters can restrict the results of the query when getting a collection of pairing profiles.

Parameter	Description	Type	Req
href	URI of the pairing profile	String	Yes
name	The short friendly name of the pairing profile. Supports partial matches.	String	
description	The long description of the pairing profile. Supports partial matches.	String	
mode	DEPRECATED AND REPLACED (Use enforcement_mode instead)	String	
enforcement_mode	Filter by mode. Has four modalities: <ul style="list-style-type: none"> • <code>idle</code>: Limited visibility. VEN does not take control of the workload IP tables(Linux) or Windows firewall (Windows). • <code>visibility_only</code>: No traffic is blocked by policy. • <code>full</code>: Segmentation rules are enforced for all inbound and outbound services. Traffic not allowed by the segmentation rule is blocked. • <code>selective</code>: Segmentation rules are enforced only for the selected inbound services when workload is within the scope of the Selective Enforcement rule. <ul style="list-style-type: none"> • Selective enforcement applies only to managed workloads; it applies neither to NEN-controlled nor to other unmanaged workloads. 	String	

Parameter	Description	Type	Req
status	VEN should be in idle state when it activates	String	
enabled	The enabled flag of the pairing profile	Boolean	
total_use_count	The number of times the pairing profile has been used	Integer	
allowed_user_per_key	The number of times the pairing profile can be used.	Integer (min 1) String	
key_lifespan	Number of seconds pairing profile keys will be valid for.	Integer (min 1) String	
last_pairing_at	Timestamp when this pairing profile was last used for a workload	String	
created_at	Timestamp when this pairing profile was first created	String date-time	
updated_at	Timestamp when this pairing profile was last updated	String date-time	
created_by		Object	
Href:	User who originally created this pairing profile	String	
updated_by		Object	
Href	User who last updated this pairing_profile	String	
is_default	Flag indicating this is default auto-created pairing profile	Boolean	
labels[]	Return only pairing profiles that have all of these labels specified as part of the pairing profile. Labels are structured in JSON as a list of lists of label HREFs.	Array	
env_label_lock	Flag that controls whether env label can be overridden from the pairing script	Boolean	
loc_label_lock	Flag that controls whether loc label can be over-	Boolean	

Parameter	Description	Type	Req
lock	ridden from the pairing script		
role_label_lock	Flag that controls whether role label can be overridden from the pairing script	Boolean	
app_label_lock	Flag that controls whether app label can be overridden from the pairing script	Boolean	
mode_lock	DEPRECATED AND REPLACED (USE /enforcement_mode_lock INSTEAD) Flag that controls whether mode can be overridden from the pairing script.		
enforcement_mode_lock	Flag that controls whether enforcement mode can be overridden from pairing script	Boolean	
visibility_level	Visibility level of the workload (DEPRECATED VALUE: 'flow_full_detail')	String	
visibility_level_lock	Flag that controls whether visibility_level can be overridden from the pairing script	Boolean	
status_lock	Flag that controls whether status can be overridden from the pairing script	Boolean	
external_data_set	External data set identifier	String Null	
external_data_set	The data source from which the resource originates. For example, if the pairing profile information is stored in an external database.	String	
external_data_reference	External data reference identifier	String Null	
enforcement_mode_lock	Flag that controls whether enforcement mode can be overridden from the pairing script	Boolean	
status_lock	Flag that controls whether status can be overridden	Boolean	

Parameter	Description	Type	Req
	from the pairing script		
agent_software_release	Agent software release associated with this pairing profile	String Null	

Examples of query parameters for filtering pairing profiles:

Filter by Name:

`/api/v2/orgs/1/pairing_profiles?name=prod_app`

Filter by Description:

`/api/v2/orgs/1/pairing_profiles?description="some description string"`

Filter by software release:

`/api/v2/orgs/1/pairing_profiles?agent_software_release="xx.x.x"`

(where the release starts with 17.2.0)

Response Body

```
{
  "href": "/orgs/1/pairing_profiles/24",
  "name": "AutoScale App",
  "description": "PP_Description",
  "total_use_count": 0,
  "enabled": true,
  "is_default": false,
  "created_at": "2016-06-08T23:13:29.019Z",
  "updated_at": "2016-06-08T23:18:36.150Z",
  "created_by": {
    "href": "/users/1"
  },
  "updated_by": {
    "href": "/users/1"
  },
  "enforcement_mode": "full",
  "key_lifespan": "unlimited",
  "allowed_uses_per_key": 1,
  "last_pairing_at": null,
  "labels": [],
}
```

```
"env_label_lock": false,
"loc_label_lock": false,
"role_label_lock": false,
"app_label_lock": false,
"mode_lock": false,
"visibility_level": "flow_summary",
"visibility_level_lock": false
},
{
  "href": "/orgs/1/pairing_profiles/27",
  "name": "Security Key Pack",
  "description": "",
  "total_use_count": 1,
  "enabled": true,
  "is_default": false,
  "created_at": "2016-06-30T18:12:22.141Z",
  "updated_at": "2016-06-30T18:13:50.668Z",
  "created_by": {
    "href": "/users/1"
  },
  "updated_by": {
    "href": "/users/57"
  },
  "mode": "illuminated",
  "key_lifespan": "unlimited",
  "allowed_uses_per_key": "unlimited",
  "last_pairing_at": "2016-06-30T18:13:50.668Z",
  "labels": [
    {
      "href": "/orgs/1/labels/153"
    },
    {
      "href": "/orgs/1/labels/154"
    },
    {
      "href": "/orgs/1/labels/155"
    },
    {
```

```
    "href": "/orgs/1/labels/156"
  }
],
"env_label_lock": true,
"loc_label_lock": true,
"role_label_lock": true,
"app_label_lock": true,
"mode_lock": true,
"visibility_level": "flow_summary",
"visibility_level_lock": true
},
```

Create a Pairing Profile

This method creates an individual pairing profile.

URI to Create a Pairing Profile

```
POST [api_version][org_href]/pairing_profiles
```

Example Request Body

```
{
  "href": "/orgs/2/pairing_profiles/12375",
  "name": "Limited Pairing",
  "description": "",
  "total_use_count": 0,
  "enabled": true,
  "is_default": false,
  "created_at": "2015-11-01T01:20:06.135Z",
  "updated_at": "2015-11-01T01:20:06.135Z",
  "created_by": {
    "href": "/users/18"
  },
  "updated_by": {
    "href": "/users/18"
  },
  "enforcement_mode": "visibility_only",
  "key_lifespan": "unlimited",
```

```
"allowed_uses_per_key": "unlimited",
"last_pairing_at": null,
"labels": [
  {
    "href": "/orgs/2/labels/6"
  },
  {
    "href": "/orgs/2/labels/14"
  },
  {
    "href": "/orgs/2/labels/8"
  },
  {
    "href": "/orgs/2/labels/12"
  }
],
"env_label_lock": false,
"loc_label_lock": false,
"role_label_lock": false,
"app_label_lock": false,
"mode_lock": true,
"visibility_level": "flow_summary",
"visibility_level_lock": true
}
```

Curl Command to Create Pairing Profile

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/pairing_profiles -H
"Content-Type:application/json" -u $KEY:'TOKEN' -d '{"href":"/orgs/2/pairing_
profiles/12375","name":"Limited Pairing","description":"","total_use_
count":0,"enabled":true,"is_default":false,"created_at":"2015-11-
01T01:20:06.135Z","updated_at":"2015-11-01T01:20:06.135Z","created_by":
{"href":"/users/18"},"updated_by":{"href":"/users/18"},"enforcement_
mode":"visibility_only","key_lifespan":"unlimited","allowed_uses_per_
key":"unlimited","last_pairing_at":null,"labels":[{"href":"/orgs/2/labels/6"},
"href":"/orgs/2/labels/14"},"href":"/orgs/2/labels/8"},"href":"/orgs/2/labels/12"}
],"env_label_lock":false,"loc_label_lock":false,"role_label_lock":false,"app_
label_lock":false,"visibility_level":"flow_summary","visibility_level_lock":true}'
```

Update a Pairing Profile

To update a pairing profile, specify its HREF, which can be obtained from getting a collection of pairing profiles.

URI to Update a Pairing Profile

```
PUT [api_version][pairing_profile_href]
```

Curl Command to Update Pairing Profile

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/pairing_profiles -H
"Accept: application/json" -u $KEY:'TOKEN' -d '{"href":"/orgs/2/pairing_
profiles/12375","name":"Limited Pairing","description":"","total_use_
count":0,"enabled":true,"is_default":false,"created_at":"2015-11-
01T01:20:06.135Z","updated_at":"2015-11-01T01:20:06.135Z","created_by":
{"href":"/users/18"},"updated_by":{"href":"/users/18"},"enforcement_
mode":"visibility_only","key_lifespan":"unlimited","allowed_uses_per_key":"one_
use","last_pairing_at":null,"labels":[{"href":"/orgs/2/labels/6"},
{"href":"/orgs/2/labels/14"},{"href":"/orgs/2/labels/8"},
{"href":"/orgs/2/labels/12"}],"env_label_lock":false,"loc_label_lock":false,"role_
label_lock":false,"app_label_lock":false,"visibility_level":"flow_
summary","visibility_level_lock":true}'
```

Delete a Pairing Profile

To delete an individual pairing profile, specify its HREF that you can obtain from a collection of pairing profiles.

URI to Delete a Pairing Profile

```
DELETE [api_version][pairing_profile_href]
```

Curl Command to Delete Pairing Profile

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/pairing_
profiles/12375 -H "Accept: application/json" -u $KEY:'TOKEN'
```

Pairing Key API Method

Functionality	HTTP	URI
Create a pairing	POST	[api_version][org_href]/pairing_profiles[pairing_profile_

Functionality	HTTP	URI
key		href]/pairing_key

Create a Pairing Key

To create a pairing key, you need a pairing profile HREF to pass as a parameter. You can obtain the pairing profile HREF from the Pairing Profile page in the PCE web console.

A pairing key is governed by the parameters configured in the pairing profile.

URI to Create a Pairing Key

Obtain the pairing key HREF from the response body returned by an API call to get a collection of pairing keys.

```
POST [api_version][pairing_key_href]/pairing_key
```

Request Body

The request body is an empty JSON object.

```
{}
```

Curl Command to Create Pairing Key

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/3/pairing_profiles/34/pairing_key -H 'Content-Type: application/json' -u $KEY:'TOKEN' -d "{}"
```

Workload Operations

The Public Stable API for workloads allows you to perform workload operations, such as create an unmanaged workload, update workload information, unpair a workload, and delete a workload.


Update Workload Information

This API allows you to update information about a workload. To make this call, you need the URI of the workload you want to update, which is returned in the form of an HREF path when you get either a single or a collection of workloads in an organization.

URI to Update an Individual Workload's Information

```
PUT [api_version][workload_href]
```

Request Properties

Property	Description	Type	Required
name	Short, friendly name of the workload.	String	Yes
description	Long description of the workload.	String	No
external_data_set	An external data set identifier. For example, if this workload information is stored in an external database.	String	No
external_data_reference	An external data set reference path. For example, if this workload information is stored in an external database.	String	No
hostname	The hostname reported by the workload.	String	Yes
service_principal_name	The Kerberos Service Principal Name (SPN). This property is only relevant if you have configured Kerberos in your environment to authenticate VEN-to-PCE communication.	String	No
distinguished_name	The X.509 Subject distinguished name, used if you want this workload to use machine authentication between VENs and other hosts. <div data-bbox="435 1308 1058 1688" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>NOTE: This property has an API exposure level of Public Experimental, which means it is not intended for production use and might change in future releases. For more information, see API Classification and Version.</p> </div>	String	No
public_ip	The public IP address of the workload.	String	No
interfaces	List of all functioning network interfaces on the workload. Properties for workload network	Array	No

Property	Description	Type	Required
	<p>interfaces:</p> <ul style="list-style-type: none"> • <code>name</code>: The interface name with a string data type • <code>link state</code>: The state of the interface connection with a string data type; one of three values: <ul style="list-style-type: none"> ◦ <code>up</code>: Interface is communicating. ◦ <code>down</code>: Interface is not communicating. ◦ <code>unknown</code>: State of the interface is unknown. • <code>address</code>: The IP address assigned to the interface with a string data type • <code>cidr_block</code>: The number of bits in the subnet (for example, /24 is 255.255.255.0) with an integer data type • <code>default_gateway_address</code>: The default IP address of the default gateway, either IPv4 or IPv6 with an integer data type • <code>friendly_name</code>: The friendly name given to the interface with a string data type 		
<code>service_provider</code>	Name of the service provider that is hosting the workload.	String	No
<code>data_center</code>	The name of the data center where the workload is being hosted.		
<code>data_center_zone</code>	The zone inside the data center hosting the workload.	String	No
<code>os_id</code>	Unique OS identifier given by Illumio to the workload.	String	No
<code>os_detail</code>	Additional descriptive information about the workload OS	String	No
<code>online</code>	Indicates whether the workload is online and can communicate with the PCE.	Boolean.	No
<code>labels</code>	Labels that are attached to the workload.	Array.	No
"agent" (VEN) properties hash:			
<code>mode</code>	This property defines the policy state of	String	No

Property	Description	Type	Required
	<p>the workload's VEN. There are three possible states:</p> <p><i>idle</i>: A state in which the VEN does not take control of the workload's iptables (Linux) or Windows firewall (Windows), but rather provides the PCE relevant details about the workload, such as the workload IP address, operating system, and geographical location.</p> <p><i>illuminated</i>: The Illuminated policy state has two variations: <i>Build</i> and <i>Test</i> (configurable in the PCE web console):</p> <ul style="list-style-type: none"> • <i>Build</i> is a state in which the VEN introspects all open ports on a workload and reports the flow of traffic between it and other workloads to the PCE. In this state, the PCE displays the flow of traffic to and from the workload. The PCE provides insight into the data center and the applications running in it. No traffic is blocked in this state. • <i>Test</i> is a state which allows you to apply all rules and visualize all the traffic that would be blocked if you placed the workloads into the enforced policy state. No traffic is blocked in this state. <p><i>enforced</i>: A state in which all rules definitions and restrictions are enforced.</p>		
<p><code>log_traffic</code> (deprecated)</p>	<p>If set to True, then traffic flows from the VEN are logged.</p>	<p>Boolean</p>	<p>No</p>
<p><code>visibility_level</code></p>	<p>The amount of data the VEN collects and reports to the PCE from a workload in the enforced mode (policy state), so you can control resource demands on workloads. The higher levels of detail are useful for</p>		

Property	Description	Type	Required
	<p>visualizing traffic flows in the Illumination map inside the PCE web console. If this parameter is not set, then VEN visibility level is set to <code>flow_summary</code>.</p> <ul style="list-style-type: none"> <code>flow_summary</code>: (“High Detail” in the PCE web console) The VEN collects traffic connection details (source IP, destination IP, protocol, and source and destination port) for both allowed and blocked connections. This option creates traffic links in the Illumination map and is typically used during the building and testing phase of your security policy. <code>flow_drops</code>: (“Less Detail” in the PCE web console.) The VEN only collects traffic connection details (source IP, destination IP, protocol, and source and destination port) for blocked connections. This option provides less detail for Illumination but demands fewer system resources from a workload and is typically used for policy enforcement. <code>flow_off</code>: (“No Detail” in the PCE web console.) The VEN does not collect any details about traffic connections. This option provides no Illumination detail and demands the least amount of resources from workloads. This mode is useful when you are satisfied with the rules that have been created and do not need additional overhead from observing workload communication. 		

Example Payload

This example shows what the JSON payload looks like for changing the policy state (called `mode` in the API) of a workload from its current state to enforced.

```
{"agent":{"config":{"mode":"enforced","log_traffic":true}}}
```

Curl Command to Update Workload

A workload state can be build, test, or enforced. This example shows how to use curl to update a workload policy state from its current state to enforced.

This example assumes that you want to update the state of a single workload in an organization. You can obtain an organization ID when you use the Users API to log in a user to Illumio.

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/3/workloads/043902c883d133fa
-H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"agent":{"config":
{"mode":"enforced","log_traffic":true}}}'
```

Unpair Workloads

This API allows you to unpair workloads from the PCE by uninstalling the Illumio VEN from each workload. You can unpair up to 1,000 workloads at a time.

Pairing a workload installs the Illumio VEN on a workload. Unpairing a workload uninstalls the VEN from the workload so that the workload no longer reports any information to the PCE, and the workload can no longer receive any policy information.

When you unpair workloads with this API, you can set the state for the workload's iptables (Linux) or WFP (Windows) configuration.

URI to Unpair Workloads

```
PUT [api_version][org_href]/workloads/unpair
```

Request Body

Property	Description	Type	Req
workloads	Defines the list of workloads you want to unpair. You must specify at least one workload to unpair by defining the workload HREF. You can define up to 1,000 workloads to unpair with this API.	Array	Yes
href	URI of the workload to unpair.	String	Yes
firewall_	This property allows you to determine the state of the	String	Yes

Property	Description	Type	Req
restore	<p>workload iptables (Linux) or WFP (Windows) configuration after the workload is unpaired.</p> <p>Options include:</p> <ul style="list-style-type: none"> • saved: Revert the iptables on the workload to the configuration before the VEN was installed. However, depending on how old the iptables or WFP configuration was on the workload, VEN removal could adversely impact security. • default: Apply the recommended policy, which is to uninstall the VEN and allow only port 22 SSH connections to the workload. Safest from a security viewpoint, but if this workload is running a production application, it could break because this workload will no longer allow any connections to it. • disable: Uninstall the VEN and leave all port connections on the workload open. This is the least safe from a security viewpoint. If iptables or WFP configuration or Illumio were the only security being used for this workload, the workload would be opened up to anyone and become vulnerable to attack. 		

Example Payload for Unpairing Workloads

```
{
  "workloads": [
    {"href": "/orgs/7/workloads/XXXXXXXXx-9611-44aa-ae06-fXXX8903db65"},
    {"href": "/orgs/7/workloads/xxxxxxxx-9611-xxxx-ae06-f7bXXX03db71"}
  ],
}
```

```
"firewall_restore":"saved"  
}
```

Curl Command for Unpairing Workload

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/3/workloads/unpair -H  
"Content-Type:application/json" -u $KEY:$TOKEN -d '{"workloads": [{"href":  
"/orgs/7/workloads/xxxxxxxx-9611-44aa-ae06-fXXX8903db65", "href":  
"/orgs/7/workloads/xxxxxxxx-9611-xxxx-ae06-f7bXXX03db71"}], "firewall_restore":  
"default"}'
```

Mark Workload as Suspended

You can use this API to mark a workload VEN as either suspended or unsuspended.

URI to Mark a Workload VEN as Suspended or Unsuspended

```
PUT [api_version][workload_href]
```

Example Payload

This example shows what the JSON payload looks like for marking a workload VEN as suspended, with the status property for the agent (the VEN) set to suspended.



NOTE:

To mark a workload VEN as unsuspended, use the same JSON body but replace suspend with unsuspend.

```
{  
  "agent": {  
    "status": {  
      "status": "suspended"  
    }  
  }  
}
```

Curl Command to Mark Workload as Suspended

This example shows you how to use curl to mark a workload VEN as suspended.

This example assumes that you want to mark a single workload VEN as suspended. You can obtain an organization ID when you use the Users API to log in a user to Illumio.

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/3/workloads/043902c883d133 -
H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"agent":{"status":
{"status":"suspended"}}}'
```

Create an Unmanaged Workload


The Unmanaged Workload API enables you to create a workload without installing the VEN on it. This API is commonly used if you are using Kerberos authentication between the VEN and the PCE.

URI to Create an Unmanaged Workload

```
POST [api_version][org_href]/workloads
```

Request Body

Property	Description	Type	Required
name	Short, friendly name of the workload.	String	Yes
description	Long description of the workload.	String	No
external_data_set	An external data set identifier. For example, if this workload information is stored in an external database.	String	No
external_data_reference	An external data set reference path. For example, if this workload information is stored in an external database.	String	No
hostname	The hostname reported by the workload.	String	Yes
service_principal_name	The Kerberos Service Principal Name (SPN). This property is only relevant if you have configured Kerberos in your environment to authenticate VEN-to-PCE communication.	String	No
distinguished_name	The X.509 Subject distinguished name, used if you want this workload to use machine authentication between VENs and other hosts.	String	No

Property	Description	Type	Required
	 <p>NOTE: This property has an API exposure level of Public Experimental, which means it is not intended for production use and might change in future releases. For more information, see API Classification and Version.</p>		
public_ip	The public IP address of the workload.	String	No
interfaces	<p>List of all functioning network interfaces on the workload.</p> <p>Properties for workload network interfaces:</p> <ul style="list-style-type: none"> • name: The interface name with a string data type • link state: The state of the interface connection with a string data type; one of three values: <ul style="list-style-type: none"> ◦ up: Interface is communicating. ◦ down: Interface is not communicating. ◦ unknown: State of the interface is unknown. • address: The IP address assigned to the interface with a string data type • cidr_block: The number of bits in the subnet (for example, /24 is 255.255.255.0) with an integer data type • default_gateway_address: The default IP address of the default gateway, either IPv4 or IPv6 with an integer data type • friendly_name: The friendly name given to the interface with a string data type 	Array	No
service_provider	Name of the service provider that is hosting the workload.	String	No

Property	Description	Type	Required
data_center	The name of the data center where the workload is being hosted.		
data_center_zone	The zone inside the data center hosting the workload.	String	No
os_id	Unique OS identifier given by Illumio to the workload.	String	No
os_detail	Additional descriptive information about the workload OS	String	No
online	Indicates whether the workload is online and can communicate with the PCE.	Boolean.	No
labels	Labels that are attached to the workload.	Array.	No
"agent" (VEN) properties hash:			
mode	<p>This property defines the policy state of the workload's VEN. There are three possible states:</p> <p><i>idle</i>: A state in which the VEN does not take control of the workload's iptables (Linux) or Windows firewall (Windows), but rather provides the PCE relevant details about the workload, such as the workload IP address, operating system, and geographical location.</p> <p><i>illuminated</i>: The Illuminated policy state has two variations: <i>Build</i> and <i>Test</i> (configurable in the PCE web console):</p> <ul style="list-style-type: none"> • <i>Build</i> is a state in which the VEN introspects all open ports on a workload and reports the flow of traffic between it and other workloads to the PCE. In this state, the PCE displays the flow of traffic to and from the workload. The PCE provides insight into the data center and the applications running in it. No traffic is blocked in this state. • <i>Test</i> is a state which allows you to apply 	String	No

Property	Description	Type	Required
	<p>all rules and visualize all the traffic that would be blocked if you placed the workloads into the enforced policy state. No traffic is blocked in this state.</p> <p>enforced: A state in which all rules definitions and restrictions are enforced.</p>		
log_traffic (deprecated)	If set to True, then traffic flows from the VEN are logged.	Boolean	No
visibility_level	<p>The amount of data the VEN collects and reports to the PCE from a workload in the enforced mode (policy state), so you can control resource demands on workloads.</p> <p>The higher levels of detail are useful for visualizing traffic flows in the Illumination map inside the PCE web console. If this parameter is not set, then VEN visibility level is set to <code>flow_summary</code>.</p> <ul style="list-style-type: none"> <code>flow_summary</code>: (“High Detail” in the PCE web console) The VEN collects traffic connection details (source IP, destination IP, protocol, and source and destination port) for both allowed and blocked connections. This option creates traffic links in the Illumination map and is typically used during the building and testing phase of your security policy. <code>flow_drops</code>: (“Less Detail” in the PCE web console.) The VEN only collects traffic connection details (source IP, destination IP, protocol, and source and destination port) for blocked connections. This option provides less detail for Illumination but demands fewer system resources from a workload and is typically used for policy enforcement. 		

Property	Description	Type	Required
	<ul style="list-style-type: none"><code>flow_off</code>: (“No Detail” in the PCE web console.) The VEN does not collect any details about traffic connections. This option provides no Illumination detail and demands the least amount of resources from workloads. This mode is useful when you are satisfied with the rules that have been created and do not need additional overhead from observing workload communication.		

Example Payload

For example, to create an unmanaged workload by providing a name, hostname, public IP address, and its Kerberos Service Principal Name, construct the JSON payload as follows:

```
{
  "name": "web_tier1",
  "hostname": "web_workload1.example.com",
  "public_ip": "10.10.10.10",
  "service_principal_name": "my_company-device-auth/web_workload1.example.com",
}
```

Curl Command to Create Unmanaged Workload

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/4/workloads -H
"Content-Type: application/json" -u $KEY:$TOKEN -d '{"name": "web_tier1",
"hostname": "web_workload1.example.com", "public_ip": "10.10.10.10", "service_
principal_name": "my_company-device-auth/web_workload1.example.com"}'
```

Delete a Workload Record

If you have unpaired a workload, you can use this API to delete the workload’s record from the PCE.

URI to Delete a Workload Record

```
DELETE [api_version][workload_href]
```

Workload Interfaces

The Public Stable API allows you to get network interface information from a workload, for either all interfaces on a workload or an individual interface. You can also configure (create) or delete an individual network interface configuration.

Workload Network Interfaces API Methods

Functionality	HTTP	URI
Get a collection of all network interfaces for a workload	GET	[api_version][workload_href]/interfaces
Get information about an individual network interface for a workload	GET	[api_version][workload_href]/interfaces/:name
Create a network interface configuration for an individual workload	POST	[api_version][workload_href]/interfaces
Delete a workload network interface configuration	DELETE	[api_version][workload_href]/interfaces/:name

Get Workload HREF and Interface Names

Before you can get network interface information for a workload, you need to know the workload's HREF, and then the name of the interface. To obtain a workload's HREF, use the Workloads API to get a collection of workloads.

In the response for getting workloads, each workload is identified by an HREF, and each network interface is identified by its name:

```
{
  "href": "/orgs/2/workloads/3e3e17ce-XXXX-42b4-XXXX-1d4d3328b342",
  "deleted": false,
  "name": standalone_dev,
  "description": null,
  "hostname": "hrm-db-prod1",
  "service_principal_name": null,
  "public_ip": "xxx.0.2.10",
  "external_data_set": null,
  "external_data_reference": null,
  "interfaces": [
    {
      "name": "eth0.public",
      "address": "xxx.0.2.10",
```

```

        "cidr_block": 32,
        "default_gateway_address": null,
        "link_state": "up",
        "network_id": 3,
        "network_detection_mode": "manual",
        "friendly_name": null
    },
    {
        "name": "eth0",
        "address": "xx.10.10.10",
        "cidr_block": null,
        "default_gateway_address": "xx.0.10.10",
        "link_state": "up",
        "network_id": 3,
        "network_detection_mode": "single_private_brn",
        "friendly_name": "Friendly eth0"
    }
],
....

```

Get Workload Network Interface

This API allows you to get information about one or all of the interfaces on a workload. You can retrieve workload interface information such as its connectivity (up, down, unknown), interface IP address, number of bits in the subnet, the IP address of the default gateway, and the associated network.

URI to Get a Collection of a Workload's Network Interfaces

```
GET [api_version][workload_href]/interfaces
```

URI to Get an Individual Workload Network Interface

```
GET [api_version][workload_href]/interfaces/:name
```

Query Parameters

Parameter	Description	Type
name	Name of the network interface on the workload.	String

Response Properties

Property	Description	Type
name	Interface name.	String
cidr_block	The number of bits in the subnet (for example, /24 is 255.255.255.0).	Integer
link_state	State of the interface connection, which is one of three values: <ul style="list-style-type: none"> up: Interface is communicating. down: Interface is not communicating. unknown: State of the interface is unknown. 	String
network_id	ID of the network hosting the workload.	
network_detection_mode	Advanced network configuration parameter for the PCE.	String
address	The IP address assigned to the interface.	String
ip_version	The address type version: <ul style="list-style-type: none"> 4: Address is an IPv4 address. 6: Address is an IPv6 address. 0: No IP address. 	Integer
default_gateway_address	The default IP address of the default gateway.	Integer
friendly_name	Friendly name given to the interface.	String

Curl Command Get Collection of Interface Statuses

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/3/workloads/12/interface_statuses -H "Accept: application/json" -u $KEY:$TOKEN
```

Create Workload Network Interface

After you have obtained a workload's HREF, you can use that HREF to make an API call that creates a new network interface configuration for the workload.

URI to Create a Workload Network Interface Configuration

```
POST [api_version][workload_href]/interfaces
```

Request Properties

Property	Description	Type	Required
name	Interface name.	String	Yes
link_state	State of the interface connection, which is one of three values: <ul style="list-style-type: none"> • up: Interface is communicating. • down: Interface is not communicating. • unknown: State of the interface is unknown. 	String	Yes
address	The IP address assigned to the interface.	String	No
default_gateway_address	The default IP address of the default gateway.	Integer	No
friendly_name	Friendly name given to the interface.	String	No

Request Body

```
{
  "name": "eth0.public",
  "address": "192.0.2.0",
  "cidr_block": 32,
  "default_gateway_address": 255.255.255.0,
  "link_state": "up",
}
```

Curl Command Create Network Interface

```
curl -i -X POST https://pce.my-company.com:8443/api/v2/orgs/2/workloads/xxxxxxx-c4e9-44e7-8a31-e86acf6b276c/interfaces -H "Content-Type: application/json" -u $KEY:$TOKEN -d '{"name": "eth0.public", "address": "192.0.2.0", "cidr_block": "32", "default_gateway_address": "255.255.255.0", "link_state": "up"}'
```

Delete Workload Network Interface

To delete a workload's network interface, you need the workload HREF and the name of the workload's network interface. Both values can be obtained in the response when you get an individual workload or a collection of workloads.

Once you have obtained a workload's HREF and the network interface name, you can use those values to make an API call that deletes the specified network interface from the workload.

URI to Delete a Workload Network Interface Configuration

```
DELETE [api_version][workload_href]/interfaces/:name
```

Curl Command Delete Network Interface

```
curl -i -X DELETE https://pce.my-company.com:8443/api/v2/orgs/2/workloads/xxxxxxx-c4e9-44e7-8a31-e86acf6b276c/interfaces/eth0 -H "Accept: application/json" -u $KEY:$TOKEN
```

Workload Settings

This Public Experimental API allows you to get network interface information from a workload, for either all interfaces on a workload or an individual interface. You can also configure (create) or delete an individual network interface configuration.

Get Workload Settings

If the VEN doesn't send a heartbeat to the PCE within the value specified in `workload_disconnected_timeout_seconds`, the workload associated with the VEN is considered to be offline.

If the VEN goes down, it sends a goodbye message to the PCE. The PCE then waits for the value set in `workload_goodbye_timeout_seconds` before setting the workload to offline.

To disable `workload_disconnected_timeout_seconds` or `workload_goodbye_timeout_seconds`, set the its value to `-1`.

For parameters, properties, JSON response body, and example curl command, see "Get Workload Settings" in the [Illumio Core REST API Reference](#).

Example JSON Response Body

```
{
  "workload_disconnected_timeout_seconds": [
    {
      "scope": [
        { "href": "/api/v2/orgs/1/labels/1" },
        { "href": "/api/v2/orgs/1/labels/3" }
      ],
      "value": -1 # never
    },
  ],
}
```



```
{
  "scope": [ ], # Empty scope is the default. Each setting must have a
  default.
  "value": 7200 # customer-supplied default
},
"workload_goodbye_timeout_seconds": [
  {
    "scope": [
      { "href": "/api/v2/orgs/1/labels/2" },
      { "href": "/api/v2/orgs/1/labels/4" }
    ],
    "value": 60
  },
  {
    "scope": [ ], # Empty scope is the default. Each setting must have a
    default.
    "value": 0 # customer-supplied default
  }
]
}
```

Update Workload Settings

This API updates workload settings. For parameters, properties, JSON request body, and example curl command, see "Update Workload Settings" in the [Illumio Core REST API Reference](#).

Example JSON Request Body to Update Both Workload Settings

```
{
  "workload_disconnected_timeout_seconds": [
    {
      "scope": [ ], # Empty scope is the default. Each setting must have a
      default.
      "value": 3600 # customer-supplied default
    }
  ],
  "workload_goodbye_timeout_seconds": [
    {
```

```
"scope": [
  { "href": "/api/v2/orgs/1/labels/1" },
  { "href": "/api/v2/orgs/1/labels/3" }
],
"value": 1800
},
{
  "scope": [
    { "href": "/api/v2/orgs/1/labels/2" },
    { "href": "/api/v2/orgs/1/labels/4" }
  ],
  "value": 60
},
{
  "scope": [ ], # Empty scope is the default. Each setting must have a
default.
  "value": 0    # customer-supplied default
}
]
}
```

Example JSON Request Body to Update One Workload Setting

```
{
  "workload_goodbye_timeout_seconds": [
    {
      "scope": [
        { "href": "/api/v2/orgs/1/labels/2" },
        { "href": "/api/v2/orgs/1/labels/4" }
      ],
      "value": 120
    },
    {
      "scope": [ ], # Empty scope is the default. Each setting must have a
default.
      "value": 0    # customer-supplied default.
    }
  ]
}
```

Workload Bulk Operations

This Public Experimental API supports “bulk” operations on collections of workloads that create, update, or delete a collection of workloads using a single API call, instead of requiring separate API calls on individual workloads one at a time.

About Bulk Operations

When you use this API to *create* a collection of workloads, a workload record is created in the PCE in the “unmanaged” state, which means that a VEN has not been installed on the workload and no policy can be applied to the workload. To apply a policy to those workloads, pair the workloads using the pairing script located in the PCE web console, or you can install and activate the VEN on the workload using the VEN control interface.

When you use this API to *update* a collection of workloads, those workloads can be either managed or unmanaged.

When you use this API to *delete* a collection of workloads, those workloads can only be unmanaged.



NOTE:
Bulk operations are rate limited to 1,000 items per operation.

Workload Bulk Operations Methods

Functionality	HTTP	URI
Create a collection of workloads	PUT	[api_version][org_href]/workloads/bulk_create
Update a collection of workloads	PUT	[api_version][org_href]/workloads/bulk_update
Delete a collection of workloads	PUT	[api_version][org_href]/workloads/bulk_delete

Caveats for Workload Bulk Operations

Bulk operations are rate limited to 1,000 items per operation. When you create, update, or delete a collection of workloads (also referred to as “bulk” operations), you can only execute one such bulk operation at a time. This means you must wait for the first bulk operation to finish before executing a new one. If you execute a new bulk operation before the currently running operation has completed, the second operation will fail with an HTTP 429 error.

Additionally, when you perform a bulk workload operation, any policy changes caused by the operation are applied to the affected VENs after the next VEN heartbeat to the PCE.

After a bulk operation completes, *all* of your PCE VEN connectivity states show as Syncing until the next VEN heartbeat. Only affected VENs receive a policy update. VENs that are not affected by policy changes transition from Syncing to In Sync after the VENs heartbeat. This process can take up to 5 minutes.

External Data Reference IDs for Workloads

External data references are a way to add your own internal identifiers to new workloads, independent of Illumio PCE-created workload HREFs. You can add these entities when you create a collection of workloads using the `bulk_create` method, or when you create an individual workload using the public API.

This capability is useful if you want to keep a set of PCE resources in sync with your internal representation of the resources, such as a configuration management database (CMDB) that holds the “source of truth” for your workloads.

Once workloads are created with these identifiers added to their properties, you can run an asynchronous query to get all workloads through an offline job, which includes the external data references in the response.

The schema for creating and updating them includes two properties:

- `external_data_set`: Identifies the original data source of the resource.
- `external_data_reference`: A unique identifier within that data source.

These properties are UTF-8 strings with a maximum length of 255 characters each. The contents must form a unique composite key, meaning that both values of these properties are treated as a unique key. These two properties together are recognized as a unique key even if one of them is left blank or set to zero,

Create a Collection of Workloads



NOTE:

Bulk operations are rate limited to 1,000 items per operation.

URI to Create a Collection of Workloads

```
PUT [api_version][org_href]/workloads/bulk_create
```

Request Body

When you create a collection of workloads, you need to pass a JSON object request body that describes the workload details.

Although this example illustrates the request body for a single workload, you can add as many workloads as you want.

For example:

```
{
  "name": "workload 0",
  "description": "workload desc 0",
  "service_principal_name": "spn 0",
  "hostname": "workload-0.example.com",
  "public_ip": "24.1.1.1",
  "external_data_set": "cldb",
  "external_data_reference": "0",
  "interfaces": [
    {
      "name": "eth0",
      "link_state": "up",
      "address": "10.0.0.2",
      "cidr_block": 24,
      "ip_version": 4,
      "default_gateway_address": "10.0.0.1",
      "friendly_name": "wan"
    }
  ],
  "labels": [
    {
      "href": "/orgs/2/labels/1"
    },
    {
      "href": "/orgs/2/labels/9"
    }
  ],
  "service_provider": "service provider",
  "data_center": "central data center",
  "os_id": "os id 0",
  "os_detail": "os detail 0",
```

```
"online": true,  
"agent": {  
  "config": {  
    "enforcement_mode": "full",  
    "visibility_level": "flow_summary"  
  }  
}
```

Curl Command to Use Bulk Create

This curl example illustrates how to create two workloads using the `bulk_create` API.

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/workloads/bulk_create  
-H "Content-Type:application/json" -u $KEY:$TOKEN -d '[{"name":"web_  
app1","description":"workload desc 0","service_principal_name":"spn 0",  
"hostname":"workload-0.example.com","public_ip":"24.1.0.1","external_data_  
set":"cmdb", "external_data_reference":"0","interfaces":[{"name":"eth0","link_  
state":"up", "address":"10.0.0.2", "cidr_block":24,"ip_version":4,"default_gateway_  
address":"10.0.0.1","friendly_name":"wan"}], "labels":  
[{"href":"/orgs/2/labels/1"}, {"href":"/orgs/2/labels/9"}], "service_provider":  
"service provider", "data_center":"central data center", "os_id":"os id 0", "os_  
detail":"os detail 0", "online":true, "agent":{"config":{"enforcement_  
mode":"visibility_only", "visibility_level":"flow_summary"}}}]
```

Update Collection of Workloads



NOTE:

Bulk operations are rate limited to 1,000 items per operation.

This method allows you to update information about a collection of workloads. To update workload information, construct the JSON object for each workload exactly as you would for modifying one workload, but modify the properties as needed.

URI to Update a Collection of Workloads

```
PUT [api_version][org_href]/workloads/bulk_update
```

Curl Command to Bulk Update Workloads

This example shows how to update two workloads with the Bulk Update API.

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/2/workloads/bulk_update
-H "Content-Type:application/json" -u $KEY:$TOKEN -d '[{"name":"web_
app1","description":"workload desc 0","service_principal_name":"spn
0","hostname":"workload-0.example.com","public_ip":"24.1.2.1","external_data_
set":"cmdb","external_data_reference":"0","interfaces":[{"name":"eth0","link_
state":"up","address":"10.0.0.2","cidr_block":24,"ip_version":4,"default_gateway_
address":"10.0.0.1","friendly_name":"wan"}],"labels":[{"href":"/orgs/2/labels/1"},
{"href":"/orgs/2/labels/9"}],"service_provider":"service provider","data_
center":"central data center","os_id":"os id 0","os_detail":"os detail
0","online":true,"agent":{"config":{"enforcement_mode":"visibility_
only","visibility_level":"flow_summary"}}},{"name":"web_app2
0","description":"workload desc 0","service_principal_name":"spn
0","hostname":"workload-0.example.com","public_ip":"24.1.3.1","external_data_
set":"cmdb","external_data_reference":"0","interfaces":[{"name":"eth0","link_
state":"up","address":"10.0.0.2","cidr_block":24,"ip_version":4,"default_gateway_
address":"10.0.0.1","friendly_name":"wan"}],"labels":[{"href":"/orgs/2/labels/1"},
{"href":"/orgs/2/labels/9"}],"service_provider":"service provider","data_
center":"central data center","os_id":"os id 0","os_detail":"os detail
0","online":true,"agent":{"config":{"enforcement_mode":"full","visibility_
level":"flow_summary"}}}]'
```

Delete a Collection of Workloads

You can use this Public Experimental API to delete a collection of unmanaged workloads.

When you call this method, you identify each unmanaged workload to delete by its HREF. For example:

```
/orgs/7/workloads/XXXXXXX-9611-44aa-ae06-fXXX8903db65
```

Additionally, if an unmanaged workload has the following two properties, for example:

- external_data_set=cmdb
- external_data_reference=25

You can construct the HREF as a query parameter that matches the values of these two properties as they are defined on the target workload. For example:

```
/orgs/1/workloads?external_data_set=cmdb&external_data_reference=25
```



NOTE:
Both query parameters must match for the exact same parameters on the workload for the delete operation to succeed.

URI to Delete a Collection of Workloads



NOTE:
Bulk operations are rate limited to 1,000 items per operation.

```
PUT [api_version][org_href]/workloads/bulk_delete
```

Request Properties

Property	Description	Type	Required
href	The HREF of a specific workload or unmanaged workload using the external_data_set and external_data_reference query parameters.	String	Yes

Request Body

```
[
  {"href": "/orgs/1/workloads/92f4a252-68d1-40ef-8cf0-b46e4ec551r"},
  {"href": "/orgs/1/workloads/92f4a252-68d1-40ef-8cf0-b46e4ecd642ix"},
  {"href": "/orgs/1/workloads?external_data_set=cmdb&external_data_reference=25"},
  {"href": "/orgs/1/workloads/92f4a252-74d1-40ef-5af0-b46a4acd333dt"}
]
```

Curl Command to Delete Collection of Workloads

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/3/workloads/bulk_delete
-H "Accept: application/json" -u $KEY:$TOKEN '[{ "href":
"/orgs/1/workloads/92f4a252-68d1-40ef-8cf0-b46e4ec551rse" },{"href":
"/orgs/1/workloads/92f4a252-68d1-40ef-8cf0-b46e4ecd642ix" }, {"href":
"/orgs/1/workloads/92f4a252-68d1-40ef-8cf0-b46e4ecd5421q" }, {"href":
"/orgs/1/workloads/92f4a252-68d1-40ef-8cf0-b46e4ecd214dt" }, {"href":
"/orgs/1/workloads/c20efa40-c615-4fa7-b8a1-badbadbadbad" },,]'
```


Response

A successful response returns an HTTP 200 message and an array of status objects, one for each workload and each workload that failed to be deleted as requested. If all unmanaged workloads are successfully deleted, an empty array is returned.

For example, two errors are shown—one where the operation was not allowed (due to lack of permissions) and one where the workload did not exist.

```
[
  {
    "href": "/orgs/1/workloads/c20efa40-c615-4fa7-b8a1-c3af4d34c5f5",
    "errors": [{"token": "method_not_allowed_error", "message": "Not allowed"}]
  },
  {
    "href": "/orgs/1/workloads/c20efa40-c615-4fa7-b8a1-badbadbadbad",
    "errors": [{"token": "not_found_error", "message": "Not found"}]
  }
]
```

VEN Operations

Overview of VEN Suspension

The VEN Update API (PUT `[api-version][ven-href]`) allows you to mark a VEN as either suspended or unsuspended in the PCE. It does not, however, actually suspend or unsuspend the VEN.

To suspend a VEN, use the `illumio-ven-ctl` command-line tool, which isolates a VEN on a workload so that you can troubleshoot issues and determine if the VEN is the cause of any anomalous behavior.

When you suspend a VEN, the VEN informs the PCE that it is in suspended mode.

However, if the PCE does not receive this notification, you must mark the VEN as "Suspended" in the PCE web console (select the VEN and click **Edit**), or you can use this API to mark the VEN as suspended.

When you don't mark the VEN as suspended in the PCE, after one hour, the PCE assumes that the workload is offline and removes it from the policy. When you mark the VEN as suspended, that VEN's workload is still included in the policy of other workloads.

When a VEN is suspended:

- The VEN still appears in the PCE in the VEN list page.
- The VEN's host cannot be unpaired.
- An organization event (`server_suspended`) is logged. This event is exportable to CEF/LEEF and has the severity of WARNING.
- Heartbeats or other communications are not expected, but when received, all communications are logged by the PCE.
- If the PCE is rebooted, the VEN remains suspended.
- Any custom iptables rules are removed, and you need to reconfigure them manually.

When a VEN is unsuspending:

- The PCE is informed that the VEN is no longer suspended and is able to receive policy from the PCE. If existing rules affect the unsuspending workload, the PCE reprograms those rules.
- An organization event (`server_unsuspended`) is logged. This event is exportable to CEF/LEEF and has the severity of WARNING.
- The workload reverts to the policy state it had before it was suspended.
- Custom iptables rules are configured back into the iptables.

VEN upgrade APIs allow you to specify an array of VEN hrefs to upgrade to a specific version instead of a list of agent ID's.

Rules when validating with the VEN upgrade APIs are as follows:

- No downgrades are allowed
- Users cannot upgrade to a VEN version higher than the PCE version
- No AIX, Solaris, or C-VEN are allowed
- Users can only upgrade VENs paired to the same region
- Only workload managers can upgrade VENs, and they can only upgrade VENs within their scope.

VEN API Methods

In addition to the page in the PCE web console that lists all VENs and shows the details of a single VEN, there is a Public Experimental API for getting VEN collections and VEN instances. Other new APIs support VEN filtering in the PCE web console, and update and unpair VENs.

VEN Methods	HTTP and URI
Get the collection of all VENs	GET [api_version][org_href]/vens/
Get details on a VEN instance	GET [api_version][org_href]/vens/ven_id
Use to get the default release without iterating through the whole collection.	GET [api_version][org_href]/software/vens/default
Support VEN filtering in the PCE web console	GET [api_version][org_href]/vens/autocomplete GET [api_version][org_href]/vens/-facets
To set the target_pce_fqdn on a VEN	PUT [api_version][org_href]/vens/ven_id
Update a VEN	PUT [api_version][org_href]/vens/update
Upgrade a VEN. This API accepts a list of hrefs instead of agent_ids. The upgrade endpoint falls under the /vens/ resource instead of the /software/ resource.	PUT [api_version][org_href]vens/upgrade
Unpair a VEN: trigger the unpairing of one or more VENs Uses the same schema as /api/v1/orgs/:xorg_id/workloads/:workload_id/unpair	PUT [api_version][org_href]/vens/unpair
Provided so that users can set the default version for VEN pairing.	PUT [api_version][org_href]/software/vens/default

Parameters for GET

Parameter	Description	Type
name	Friendly name for the VEN.	String
hostname	The hostname of the host managed by the VEN.	String
description	Description of VEN(s) to return. Supports partial matches	
uid	The unique ID of the host managed by the VEN	String
os_id	Operating System of workload(s) to return. Supports partial matches.	String
os_detail	Additional OS details from the host managed by the VEN	

Parameter	Description	Type
version	Software version of the VEN.	String
status	The current status of the VEN. Options are: "active", "suspended", "uninstalled"	String
activation_type	The method by which the VEN was activated. Options are: "pairing_key", "kerberos", "certificate"	String
active_pce_fqdn	The FQDN of the PCE that the VEN last connected to	String
target_pce_fqdn	The FQDN of the PCE that the VEN uses for future connections	String
labels	Labels assigned to the host managed by the VEN.	String
ip_address	IP address of VEN(s) to return. Supports partial matches	String
interfaces	Network interfaces of the host managed by the VEN.	Array
container_cluster	Array of container cluster URIs, encoded as a JSON string	String
security_policy_applied_at	Last reported time when policy was applied to the workload (UTC), only present in expanded representations.	date-time
security_policy_received_at	Last reported time when policy was received by the workload (UTC), only present in expanded representations.	date-time
enforcement_mode	Policy enforcement mode, only present in expanded representations. Options are: "idle", "visibility_only", "full", "selective"	String
visibility_level	Visibility level of the workload, only present in expanded representations.	String
secure_connect	Property: <code>matching_issuer_name</code> . Issuer name match criteria for certificate used during establishing secure connections. <code>matching_issuer_name</code> : Issuer name match criteria for certificate used while establishing secure connections.	Object String
last_heartbeat_at	The last time (rfc3339 timestamp) a heartbeat was received from this VEN	Hash
last_goodbye_at	The time (rfc3339 timestamp) of the last goodbye from the VEN	Hash
created_at	The time (rfc3339 timestamp) at which this VEN was created	date-time

Parameter	Description	Type
created_by	The user who created this VEN	
updated_at	The time (rfc3339 timestamp) at which this VEN was last updated	date-time
updated_by	The URI of the user who last updated this VEN	date-time

Curl Commands for GET VENs

To get a collection of VENs that have a specific label applied to them, take a label string that was returned when you got a collection of VENs, and then add a query parameter to GET all VENs with that specific label.

Curl Command to Get VENs with a Specific Label

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/vens?labels="
[[/orgs/2/labels/1642]]" -H "Accept: application/json" -u $KEY:$TOKEN
```

To restrict the type of VENs you want returned and set a limit on how many results you want returned, use the relevant query parameters. For example, you might want to get a collection of no more than 50 VENs running CentOS 6.3 that have an active status.

Curl Command to Get VENs using other Query Parameters

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/2/vens?os_id=centos-
x86_64-6.3&max_results=50&status=active -H "Accept: application/json"-u
$KEY:$TOKEN
```

Curl Commands for Unpairing and Suspending VENs

Instead of unpairing and suspending workloads, use the new VEN APIs to unpair and suspend VENs.

Parameters for PUT

Parameter	Description	Type
vens	VENs to unpair (required)	Array
href	URI of VEN to unpair (required)	String
firewall_ restore	The strategy to use to restore the firewall state after the VEN is uninstalled: Options are: saved, default, and disable. The default is: default.	String

Curl Command for Unpairing VENs

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/3/vens/unpair -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"vens": [{"href": "/orgs/7/vens/xxxxxxxx-9611-44aa-ae06-fXXX8903db65"}, {"href": "/orgs/7/vens/xxxxxxxx-9611-xxxx-ae06-f7bXXX03db71"}], "firewall_restore": "default"}'
```

Curl Command to Mark VEN as Suspended

```
curl -i -X PUT https://pce.my-company.com/api/v2/orgs/3/vens/xxxxxxxx-9611-xxxx-ae06-f7bXXX03db71 -H "Content-Type:application/json" -u $KEY:$TOKEN -d '{"status": "suspended"}'
```

- PUT /api/v2/orgs/:xorg_id/vens/unpair [Schema change](#)

Agents on Workloads

This Public Experimental API gets and updates an agent on a workload.



WARNING:

The term Agent has been deprecated and replaced by VEN. It will be removed in future releases.

Instead of this deprecated API, see the information in [VEN Operations](#).

Agents API Methods

Functionality	HTTP	URI
Get an individual agent	GET	[api_version][agent_href]
Update an agent	PUT	[api_version][agent_href]/update

Get an Individual Agent

This API returns an agent record.

Curl Command to Get an Agent




NOTE:

To obtain the agent ID, make a call to a managed workload (a workload associated with a VEN) GET /workloads/workload_id. To get all managed workloads, make a call to GET /workloads?managed=true.

```
curl -i -X GET https://pce.my-company.com:8443/api/v2/orgs/7/agents/12345 -H
"Accept: application/json" -u $KEY:$TOKEN
```

Path Parameters

Parameter	Description	Type
org_id	The ID of the organization.	String
agent_id	<p>The agent ID.</p> <p>To obtain the agent ID, make a call to a managed workload (a workload associated with a VEN) GET /workloads/workload_id.</p> <p>To get all managed workloads, make a call to GET /workloads?managed=true.</p> <div data-bbox="402 1102 1279 1255" style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  <p>NOTE: The agent_id is the integer at the end of the agent href. Example: "href":"/orgs/7/agents/12345"</p> </div>	Integer

Example Response Body

```
{
  "href": "/orgs/7/agents/12345",
  "org_id": 1,
  "name": null,
  "description": null,
  "uid": "xxxxx-xxxxx--8ef6-c0fb3fea3cf5",
  "last_heartbeat_on": "2018-09-08T01:12:00.999Z",
  "uptime_seconds": 9944537,
  "hostname": "my-hostname",
  "agent_version": "16.9.0",
  "public_ip": "xx.xxx.xxx.xxx",
  "status": "active",
}
```

```
"online": true,
"fw_rules_generation_actual": 1,
"service_provider": "my-datacenter-provider.com",
"data_center": "123.my-datacenter.com",
"data_center_zone": "3",
"instance_id": "pi@xxxxxxxx93816",
"fw_config_current?": true,
"managed_since": "2018-06-13T03:23:00.000Z",
"os_id": "ubuntu-x86_64-xenial",
"os_detail": "4.4.0-97-generic #120-Ubuntu SMP Tue Sep 19 17:28:18 UTC 2017
(Ubuntu 16.04.1 LTS)",
"visibility_level": "flow_summary",
"created_at": "2018-06-13T03:23:00.000Z",
"updated_at": "2018-09-08T01:02:00.000Z",
"created_by": {
  "href": "/orgs/7/agents/12345"
},
"updated_by": null,
"service_report": {
  "uptime_seconds": 9887714,
  "created_at": "2018-09-07T21:05:44.825Z",
  "open_service_ports": [
    {
      "protocol": 17,
      "address": "0.0.0.0",
      "port": 67,
      "process_name": "dhcpcd",
      "user": "root",
      "package": null,
      "win_service_name": null
    },
    {
      "protocol": 6,
      "address": "0.0.0.0",
      "port": 53,
      "process_name": "bind",
      "user": "root",
      "package": null,
```



```
    "win_service_name": null
  },
  {
    "protocol": 17,
    "address": "0.0.0.0",
    "port": 123,
    "process_name": "ntpd",
    "user": "root",
    "package": null,
    "win_service_name": null
  }
]
},
"labels": [
  {
    "href": "/orgs/7/labels/2010"
  },
  {
    "href": "/orgs/7/labels/300"
  },
  {
    "href": "/orgs/7/labels/2000"
  },
  {
    "href": "/orgs/7/labels/260"
  }
],
"mode": "illuminated",
"target_pce_fqdn": null,
"active_pce_fqdn": null
}
```

Update an Agent

This API updates the agent `target_pce_fqdn` parameter to point to the FQDN of a Supercluster or a PCE that is a member of a Supercluster.

URI to Update an Agent “target_pce_fqdn” Parameter

```
PUT [api_version][agent_href]/update
```

Curl Command to Update an Agent

```
curl -i -X PUT https://pce.my-company.com:8443/api/v2/orgs/7/agents/12345 -H
"Content-Type: application/json" -u $KEY:$TOKEN -d '{"target_pce_fqdn":
"my.supercluster.pce.my-company.com"}'
```

Request Body

Property	Description	Type
target_pce_fqdn	FQDN of the target Supercluster or the target PCE that is a member of a Supercluster.	String

Example Request Body

```
{
  "target_pce_fqdn": "my.supercluster.pce.my-company.com"
}
```

Blocked Traffic to and from Workloads

This Public Experimental API was used to identify unauthorized traffic to or from workloads. It would get a list of blocked or potentially blocked traffic between workloads and other entities managed by the PCE.



WARNING:

In the 19.1.0 release, blocked traffic was marked for deprecation and is now turned off by default.

The functionality previously provided by blocked traffic API is available in [Explorer](#).

The Blocked Traffic page continues to work, and when you configure the Explorer feature, this page uses the Explorer API to get the data from PCE.

Filtering and Aggregating Traffic

This Public Experimental API method allows you to handle broadcast and multicast traffic better, save storage in the traffic database, and reduce the stress of the whole data pipeline.

Windows-heavy environments can have a large amount of broadcast or multicast traffic, which can be as much as 50% in syslog data and 30% in traffic data. Because some broadcast and multicast data might not be useful for writing policies, this API

provides a function to filter out or aggregate the broadcast and multicast traffic that is not useful.



NOTE:
This API is implemented in Supercluster.



NOTE:
Only administrators and users with appropriate privileges can make filtering changes.

Traffic Collector API Methods

Use these methods to get, create, update, or delete a traffic collector.

Functionality	HTTP	URI
Get a traffic collector collection	GET	GET/api/v2/orgs/:xorg_id/settings/traffic_collector
Get a specific collector instance	GET	GET/api/v2/orgs/:xorg_id/settings/traffic_collector/:uuid
Create a traffic collector	POST	POST /api/v2/orgs/:xorg_id/-settings/traffic_collector
Update a specific traffic collector instance	PUT	PUT /api/v2/orgs/:xorg_id/settings/traffic_collector/:uuid
Delete a specific traffic collector instance	DELETE	DELETE /api/v2/orgs/:xorg_id/-settings/traffic_collector/:uuid

Query Parameters

Use the following required and optional parameters for the query.

Parameters	Description	Type
org_id	Required for all methods.	Integer
transmission	Required for POST and optional for PUT. For the transmission type, choose "broadcast" or "multicast."	String
action	Required for POST and optional for PUT. Drop or aggregate the target traffic: <ul style="list-style-type: none"> If you select "drop," the PCE drops all the traffic that matches the filters you supply. The data will be lost forever. If you select "aggregate," the PCE performs aggregation 	String

Parameters	Description	Type
	<p>on broadcast traffic and multicast traffic . If one broadcast or multicast traffic flow is received by multiple workloads, all reported flows on the same traffic are aggregated into one record in the traffic database, and the destination workload information will be lost.</p> <ul style="list-style-type: none"> • PUT method will fail if you change the aggregator from “broadcast” to “multicast” because the default port and protocol will not pass the validation step. 	
target	<p>The target object has the following properties:</p> <ul style="list-style-type: none"> • dst_port: Optional • proto: Required • dst_ip: A single IP address or CIDR; optional <p>If dst_port and dst_ip are not specified for the target session, traffic is dropped on “all ips” and “all ports” by default.</p> <p>PUT method will fail If the traffic filter you want to modify has “ANY” in port or protocol field, and you want to modify other fields in this filter. The change will fail because the default port and protocol will not pass the validation step.</p>	<p>Object Integer Integer String</p>

Curl Command Examples for Traffic Collector

Broadcast Transmission and Drop Action

```
curl 'https://pce.my-company.com:8443/api/v2/orgs/1/settings/traffic_collector' -H
'Origin: https://pce.my-company.com:8443' -H 'Accept-Encoding: gzip,deflate, br' -
H 'content-type: application/json' -H 'accept: application/json' -H 'Referer:
https://pce.my-company.com:8443/' -i -u api_
1dfe2432a7b314ee6:'21c10ea1a4ad38d76ef22977e8ac45bc10839c5cc6ebffd650eae4f95dc5b36
4'--data-binary '{"transmission": "broadcast","action": "drop","target":{"proto":
17,"dst_port": 20, "dst_ip":"10.255.255"}}' --compressed
```

Multicast Transmission and Aggregate Action

```
curl 'https://pce.my-company.com:8443/api/v2/orgs/1/settings/traffic_collector' -H
'Origin: https://pce.my-company.com:8443' -H 'Accept-Encoding: gzip, deflate, br'
-H 'content-type: application/json' -H 'accept: application/json' -H 'Referer:
```

```
https://pce.my-company.com:8443/' -i -u api_
1dfe2432a7b314ee6:'21c10ea1a4ad38d76ef22977e8ac45bc10839c5cc6ebffd650eae4f95dc5b36
4'--data-binary '{"transmission": "multicast","action": "aggregate"}' --
compressed
```

Example Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "required": ["href", "transmission", "action"],
  "properties": {
    "href": {
      "description": "URI of the destination",
      "type": "string"
    },
    "transmission": {
      "description": "transmission type: broadcast/multicast",
      "type": "string",
      "enum": [
        "broadcast",
        "multicast"
      ]
    },
    "target": {
      "type": "object",
      "required": [
        "proto"
      ],
      "properties": {
        "dst_port": {
          "type": "integer"
        },
        "proto": {
          "type": "integer"
        },
        "dst_ip": {
          "type": "string",
          "description": "single ip address or CIDR"
        }
      }
    }
  }
}
```

```
        }
      }
    },
    "action":{
      "description":"drop or aggregate the target traffic",
      "type":"string",
      "enum":[
        "drop",
        "aggregate"
      ]
    }
  }
}
```