Secure Workload OpenAPIs

OpenAPI provides a REST API for Secure Workload features.

OpenAPI Authentication

OpenAPI uses a digest-based authentication scheme. The workflow is as follows:

1. Log in to the Secure Workload UI Dashboard.
2. Generate an API key and an API secret with the desired capabilities.
3. Use Secure Workload API SDK to send REST requests in JSON format.
4. To use the Python SDK, you install the SDK using `pip install tetpyclient`.
5. After the Python SDK is installed, here is some boilerplate code for instantiating the RestClient:

```
from tetpyclient import RestClient

API_ENDPOINT="https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"

# ``verify`` is an optional param to disable SSL server authentication.
# By default, cluster dashboard IP uses self signed cert after
# deployment. Hence, ``verify=False`` might be used to disable server
# authentication in SSL for API clients. If users upload their own
# certificate to cluster (from ``Platform > SSL Certificate``)
# which is signed by their enterprise CA, then server side authentication
# should be enabled; in such scenarios, in the code below, verify=False
# should be replaced with verify="path-to-CA-file"
# credentials.json looks like:
# {
#   "api_key": "<hex string>",
#   "api_secret": "<hex string>"
# }

restclient = RestClient(API_ENDPOINT,
                credentials_file='<path_to_credentials_file>/credentials.json',
                verify=False)
# followed by API calls, for example API to retrieve list of agents.
# API can be passed /openapi/v1/sensors or just /sensors.
resp = restclient.get('/sensors')
```

Generate API Key and Secret

Procedure

Step 1 In the upper right corner of Secure Workload UI, click the logged in account and choose **API Keys**.

Step 2 Click **Create API Key**.

Step 3 (Optional) Enter a description for the API key.

Step 4 Select the required capabilities for the key and secret.

Select the limited set of capabilities that are intended for using the API Key+Secret pair.

Note The availability of the API capabilities varies based on the user role.

Table 1: API Capabilities

Capability	Description
`sensor_management`	To configure and monitor status of software agents
`software_download`	To download software packages for agents or virtual appliances
`flow_inventory_query`	To query flows and inventory items in Secure Workload cluster
`user_role_scope_management`	To read, add, modify, or remove users, roles, and scopes
`user_data_upload`	To allow users to upload data for annotating flows and inventory items or upload good or bad file hashes
`app_policy_management`	To manage workspaces (applications) and enforce policies
`external_integration`	To allow integration with external systems such as vCenter and kubernetes

Table 2: API Capabilities for Site Administrators

Capability	Description
`appliance_management`	To manage Secure Workload appliance
`appliance_monitoring`	To monitor Secure Workload appliance settings and configurations (read-only)

Step 5 Click **Create**.
API key and secret are generated and must be copied to a file, and saved in a safe location. Alternatively, you can download the JSON file with the key and secret.

Note If External Auth with LDAP and LDAP Authorization are enabled, access to OpenAPI using API Keys stop because Secure Workload roles that are derived from LDAP MemberOf groups are reassessed after the user session terminates. Therefore, to ensure uninterrupted OpenAPI access, it is recommended that any user with API keys have the **Use Local Authentication** option that is enabled in the Edit User Details flow for the user.

Workspaces and Security Policies

The following pages describe the OpenAPI endpoints to manage Segmentation.

Workspaces

Workspaces (formerly known as “application workspaces” or “Applications”) are the containers for defining, analyzing and enforcing policies for the workloads in a particular scope. For more information about how they work see the Workspaces documentation. This set of APIs requires the `app_policy_management` capability associated with the API key.

Workspace Object

The workspace (“application”) JSON object is returned as a single object or an array of objects depending on the API endpoint. The object’s attributes are described below:

Attribute	Type	Description
id	string	A unique identifier for the workspace.
name	string	User specified name of the workspace.
description	string	User specified description of the workspace.
app_scope_id	string	ID of the scope with which the workspace is associated.
author	string	First and last name of the user who created the workspace.
primary	boolean	Indicates if the workspace is primary for its scope.
alternate_query_mode	boolean	Indicates if ‘dynamic mode’ is used for the workspace. In the dynamic mode, an automatic policy discovery run creates one or more candidate queries for each cluster. Default value is true.
created_at	integer	Unix timestamp of when the workspace was created.
latest_adm_version	integer	The latest adm (v*) version of the workspace.
analysis_enabled	boolean	Indicates if analysis is enabled on the workspace.
analyzed_version	integer	The analyzed p* version of the workspace.
enforcement_enabled	boolean	Indicates if enforcement is enabled for the workspace.
enforced_version	integer	The enforced p* version of the workspace.

List Applications

This endpoint will return an array of workspaces (“applications”).

```
GET /openapi/v1/applications
```

Table 3: Parameters

Name	Type	Description
app_scope_id	string	Match workspaces associated with a specific app scope.
exact_name	string	Match workspaces with exactly the value provided.

Response object: Returns an array of workspace objects.

Sample python code

```
restclient.get('/applications')
```

Retrieve a Single Workspace

This endpoint will return the requested workspace (“application”) as a single JSON object.

```
GET /openapi/v1/applications/{application_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Response object: Returns the workspace object for the specified ID.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s' % application_id)
```

Create a Workspace

This endpoint creates a workspace (“application”). It is possible to define policies by posting a JSON body containing the cluster and policy definitions.

Note If a primary workspace exists for the same scope and new policies are provided, the policies will be added as a new version to the existing workspace.

```
POST /openapi/v1/applications
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
app_scope_id	string	The scope ID to assign to the workspace.

Name	Type	Description
name	string	(optional) A name for the workspace.
description	string	(optional) A description for the workspace.
alternate_query_mode	boolean	(optional) Indicates if 'dynamic mode' is used for the workspace. In the dynamic mode, an automatic policy discovery run creates one or more candidate queries for each cluster. Default value is true.
strict_validation	boolean	(optional) Will return an error if there are unknown keys or attributes in the uploaded data. Useful for catching misspelled keys. Default value is false.
primary	string	(optional) Set to 'true' to if this workspace should be primary for the associated scope. **Default is true**

Extra optional parameters may be included describing policies to be created within the workspace.

Note The scheme corresponds to that returned during export from the UI and the **Details** endpoint.

Name	Type	Description
clusters	array of clusters	Groups of nodes to be used to define policies.
inventory_filters	array of inventory filters	Filters on datacenter assets.
absolute_policies	array of policies	Ordered policies to be created with the absolute rank.
default_policies	array of policies	Ordered policies to be created with the default rank.
catch_all_action	string	"ALLOW" or "DENY"

Cluster object attributes:

Name	Type	Description
id	string	Unique identifier to be used with policies.

Name	Type	Description
name	string	Displayed name of the cluster.
description	string	Description of the cluster.
nodes	array of nodes	Nodes or endpoints that are part of the cluster.
consistent_uuid	string	Must be unique to a given workspace. After an automatic policy discovery run, the similar/same clusters in the next version will maintain the consistent_uuid.

Node object attributes:

Name	Type	Description
ip	string	IP or subnet of the node. For example 10.0.0.0/8 or 1.2.3.4
name	string	Displayed name of the node.

Inventory Filter object attributes:

Name	Type	Description
id	string	Unique identifier to be used with policies.
name	string	Displayed name of the cluster.
query	object	JSON object representation of an inventory filter query.

Policy object attributes:

Name	Type	Description
consumer_filter_id	string	ID of a cluster, user inventory filter or app scope.
provider_filter_id	string	ID of a cluster, user inventory filter or app scope.
action	string	“ALLOW” or “DENY”
l4_params	array of l4params	List of allowed ports and protocols.

L4Params object attributes:

Name	Type	Description
proto	integer	Protocol Integer value (NULL means all protocols).

Name	Type	Description
port	array	Inclusive range of ports. For example, [80, 80] or [5000, 6000].
approved	boolean	(optional) Indicates if the policy is approved. Default is False.

Response object: Returns the newly created workspace object.

Sample python code

```
  name = 'test'
  scope_id = '5ce480cc497d4f1b4b9a9e8d'
  filter_id = '5ce480cd497d4f1b4b9a9ea4'
  application = {
      'app_scope_id': scope_id,
      'name': name,
      'absolute_policies': [
          {
       # consumer/provider filter IDs can be ID of a cluster identified during automatic
policy discovery (formerly known as ADM),
       # user inventory filter or app scope.
              'provider_filter_id': filter_id,
              'consumer_filter_id': filter_id,
              'action': 'ALLOW',
       # ALLOW policy for TCP on port 80.
              'l4_params': [
                  {
                      'proto': 6, # TCP
                      'port': [80, 80],  # port range
                  }
              ],
          }
      ],
      'catch_all_action': 'ALLOW'
  }
  restclient.post('/applications', json_body=json.dumps(application))
```

Import a New Version

Imports policies and creates a new v* version for the workspace (“application”).

`POST /openapi/v1/applications/{application_id}/import`

The parameters are the same as the create workspace endpoint.

Response object: Returns the workspace object.

Validate a Set of Policies

Validates a set of policies without creating a new version.

`POST /openapi/v1/applications/validate_policies`

An *app_scope_id* is required. The rest of the parameters are the same as the create workspace endpoint.

Response object:

Attribute	Type	Description
valid	boolean	Indicates if the policies are valid
errors	array	If invalid, details about the errors

Delete a Workspace

Removes a workspace (“application”).

```
DELETE /openapi/v1/applications/{application_id}
```

Enforcement must be disabled on the workspace before it can be deleted.

If the workspace, or its clusters, are used on by other Applications (via a Provided Service relationship) this endpoint will return `422 Unprocessable Entity`. The returned Error object will contain a `details` attribute with the count of dependent objects along with the ids of the first 10 of each type. This information can be used to locate and remove the blocking dependencies.

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Response object: None

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.delete('/applications/%s' % application_id)
```

Update a Workspace

This end point updates an existing workspace (“application”).

```
PUT /openapi/v1/applications/{application_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

The JSON query body contains the following keys

Name	Type	Description
name	string	(optional) The updated name for the workspace.
descrip	string	(optional) The updated description for the workspace.

Name	Type	Description
primary	string	(optional) Set to ‘true’ to make the workspace primary. Set to ‘false’ to make the workspace secondary.

Response object: The updated workspace object for the specified ID.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'name': 'Updated Name',
      'description': 'Updated Description',
      'primary': 'true'
}
resp = restclient.put('/applications/%s' % application_id,
                      json_body=json.dumps(req_payload))
```

Retrieve Workspace Details

This endpoint returns a full export JSON file for the workspace. This will include policy and cluster definitions.

```
GET /openapi/v1/applications/{application_id}/details
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.
version	string	(optional) A version in the form of ‘v10’ or ‘p10’, defaults to ‘latest’.

Response object: Returns the clusters and policies for the given workspace version.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
# For v* version v10 and for p* version p10
version = 'v10'
resp = restclient.get('/applications/%s/details?version=%s' % (application_id, version))
```

List Workspace Versions

This endpoint will return a list of all the versions for a given workspace.

```
GET /openapi/v1/applications/{application_id}/versions
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Name	Type	Description
created_before	integer	(optional) For pagination, set to 'created_at' of the last version from previous response.
limit	integer	(optional) Max results to return, default is 50.

Response object: An array of objects with the following attributes:

Attribute	Type	Description
version	string	A version in the form of 'v10' or 'p10'.
created_at	integer	Unix timestamp of when the workspace was created.
description	string	User provided description.
name	string	Displayed name.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
created_before = 1612325705
limit = 10
resp = restclient.get('/applications/%s/versions?created_before=%s&limit=%s' %
                      (application_id, created_before, limit))
```

Delete Workspace Version

This endpoint will remove the given version including clusters and policies. Enforced or Analyzed versions can not be deleted. If members are referenced by another workspace, through an external policy, the response will return error with a list of the references.

```
DELETE /openapi/v1/applications/{application_id}/versions/{version}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.
version	string	A version in the form of 'v10' or 'p10'.

Response object: None

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
version = 'v10'
resp = restclient.delete('/applications/%s/versions/%s' %
                         (application_id, version))
```

Compare Workspace versions

This endpoint computes the difference between the provided workspace versions. It will return the added, removed and optionally the unchanged policies. Cluster changes are included if the cluster is present in both versions, defined by a matching consistent_uuid, and the query has changed.

```
GET /openapi/v1/applications/{application_id}/version_diff
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.
base_version	string	Full version, for example ‘v10’ or ‘p10’.
draft_version	string	Full version, for example ‘v10’ or ‘p10’.
include_unchanged	boolean	Default is false. Returns unchanged policies in the response.

Response object: Returns an object with the following attributes:

Attribute	Type	Description
clusters	array	The clusters that have changed between the versions.
policies	array	The policies that have changed between the versions.

Analyze latest policies

Enable analysis on the latest set of policies in the workspace.

```
POST /openapi/v1/applications/{application_id}/enable_analysis
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Parameters: The optional JSON query body contains the following keys

Name	Type	Description
action_note	string	(optional) Reason for the publish policies action.
name	string	(optional) Name for the published policy version.

Name	Type	Description
description	string	(optional) description for the published policy version.

Response object: Returns an object with the following attributes:

Attribute	Type	Description
data_set	object	JSON object representation of the data set.
analyzed_policy_version	integer	The analyzed p* version of the workspace.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
  'action_note': 'Policy analysis',
  'name': 'Test run 1',
  'description': 'New workloads added.'
}
resp = restclient.post('/applications/%s/enable_analysis' % application_id,
                       json_body=json.dumps(req_payload))
```

Disable policy analysis on a single workspace

Disable policy analysis on the workspace.

```
POST /openapi/v1/applications/{application_id}/disable_analysis
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Response object: Returns an object with the following attributes:

Attribute	Type	Description
data_set	object	JSON object representation of the data set.
analyzed_policy_version	integer	Last analyzed p* version of the workspace.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_analysis' % application_id)
```

Enforce a single workspace

Enable enforcement on the latest set of policies in the workspace.

```
POST /openapi/v1/applications/{application_id}/enable_enforce
```

Warning New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.
version	string	(optional) The policy version to enforce.

If a version is not provided the latest policies of the workspace will be enforced. versions is preferred to be of the form ‘p*’, if just an integer is provided the corresponding ‘p*’ version will be enforced.

Response object: Returns an object with the following attributes:

Name	Type	Description
epoch	string	Unique identifier for the latest enforcement profile.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
  'version': 'p10'
}
resp = restclient.post('/applications/%s/enable_enforce' % application_id,
                       json_body=json.dumps(req_payload))
```

Disable enforcement for a single workspace

Disable enforcement on the workspace.

```
POST /openapi/v1/applications/{application_id}/disable_enforce
```

Warning New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Response object: Returns an object with the following attributes:

Name	Type	Description
epoch	string	Unique identifier for the latest enforcement profile.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_enforce' %
                       application_id)
```

Initiate Automatic Policy Discovery

Automatically discover policies for the workspace. (Formerly known as “submitting an ADM run”).

```
POST /openapi/v1/applications/{application_id}/submit_run
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Parameters: The JSON query body contains the following keys

Name	Type	Description
start_time	string	Start time of the input time interval for an automatic policy discovery run.
end_time	string	End time of the input time interval for an automatic policy discovery run.
clustering_granularity	string	(optional) Clustering Granularity allows the user to have a control on the size of the clusters generated by automatic policy discovery. Expected values: VERY_FINE, FINE, MEDIUM, COARSE, or VERY_COARSE
port_generalization	string	(optional) Port Generalization controls the level of statistical significance required when performing port generalization. Expected values: DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, or VERY_AGGRESSIVE

Name	Type	Description
policy_compression	string	(optional) Policy Compression when enabled, policies that are sufficiently frequent, i.e. they use the same provider port, among the generated clusters inside a workspace may be 'factored out' to the parent, that is, replaced with one or more policies applicable to the entire parent scope. Expected values: DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, or VERY_AGGRESSIVE
auto_accept_policy_connectors	boolean	(optional) Auto accept policy connectors any outgoing policy requests created during the automatic policy discovery will be auto accepted.
enable_exclusion_filter	boolean	(optional) Enable exclusion filter option provides the flexibility to ignore all conversations matching any of the user-defined exclusion filters (if any). For more information, see Exclusion Filters.
enable_default_exclusion_filter	boolean	(optional) Enable default exclusion filter option provides the flexibility to ignore all conversations matching any of the default exclusion filters (if any). For more information, see Default Exclusion Filters for more info.
enable_service_discovery	boolean	(optional) When Enable service discovery on agent is set, ephemeral port-range information about services present on the agent node are reported. Policies are then generated based on the reported port-range information.
carry_over_policies	boolean	(optional) When Carry over Approved Policies is set, all the policies that are marked as approved by the user via UI or OpenAPI will be preserved.

Name	Type	Description
skip_clustering	boolean	(optional) When Skip clustering is set, no new clusters are generated, and policies are generated from any existing approved clusters or inventory filters and otherwise involve all workloads in the scope.
deep_policy_generation	boolean	(Optional) You can generate policies for a branch of the scope tree rather than for a single scope. For more information, see Discover Policies for One Scope or for a Branch of the Scope Tree and subtopic.
use_default_config	boolean	(optional) When this option is set, automatic policy discovery will use the Default Policy Discovery Config instead of the previous run config. For more information, see Default Policy Discovery Config.

Note Unspecified optional parameter default values will be taken from the previous automatic policy discovery run config if automatic policy discovery was performed earlier in the workspace or else the default values will be taken from the Default Policy Discovery Config.

Response object: Returns an object with the following attributes:

Name	Type	Description
message	string	Message about success or failure of automatic policy discovery run.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'start_time': '2020-09-17T10:00:00-0700',
      'end_time': '2020-09-17T11:00:00-0700',
      # Optional Parameters.
      'clustering_granularity': 'FINE',
      'port_generalization': 'AGGRESSIVE',
      'policy_compression': 'AGGRESSIVE',
      'auto_accept_policy_connectors': False,
      'enable_exclusion_filter': True,
      'enable_default_exclusion_filter': True,
      'enable_service_discovery': True,
      'carry_over_policies': True,
      'skip_clustering': False,
      'deep_policy_generation': True,
      'use_default_config': False
```

```
}
resp = restclient.post('/applications/%s/submit_run' % application_id,
                       json_body=json.dumps(req_payload))
```

Get Status of a Policy Discovery Run

Query the status of an automatic policy discovery run in the workspace.

```
GET /openapi/v1/applications/{application_id}/adm_run_status
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

Response object: Returns an object with the following attributes:

Name	Type	Description
status	string	Status of the automatic policy discovery run. Values: PENDING, COMPLETE, or FAILED

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/applications/%s/adm_run_status' % application_id)
```

Policies

This set of APIs can be used to manage add, edit or delete Policies. `version` parameter is required for create and update catch all actions. They require the `user_role_scope_management` capability associated with the API key.

Policy object

The policy object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the policy.
application_id	string	The id for the workspace to which the policy belongs.
consumer_filter_id	string	ID of a defined filter. Currently, any cluster, user defined filter or scope can be used as the consumer of a policy.

Attribute	Type	Description
provider_filter_id	string	ID of a defined filter. Currently, any cluster, user defined filter or scope can be used as the provider of a policy.
version	string	Indicates the version of the workspace to which the policy belongs.
rank	string	Policy rank, possible values: DEFAULT, ABSOLUTE or CATCHALL.
policy_action	string	Possible values can be ALLOW or DENY. Indicates whether traffic should be al- lowed or dropped for the given service port/protocol between the consumer and provider.
priority	integer	Used to sort policy.
l4_params	array of l4params	List of allowed ports and protocols.

L4Params object attributes:

Name	Type	Description
proto	integer	Protocol Integer value (NULL means all protocols).
port	array	Inclusive range of ports. eg [80, 80] or [5000, 6000].
description	string	Short string about this proto and port.
approved	boolean	If the policy has been approved by the user.

Get Policies

This endpoint returns a list of policies in a particular workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/policies
```

Parameters: The request URL contains the following parameters

Name	Type	Description
version	string	Indicates the version of the workspace from which to get the policies.
consumer_filter_id	string	(optional) Filters the output by the consumer filter id.
provider_filter_id	string	(optional) Filters the output by the consumer filter id.

Policy IDs can change from version to version. To obtain the list of policies from a published version, the version number should be prefixed with a 'p'. For example, to fetch all the policies in published version 3 we can perform a request such as:

```
GET /openapi/v1/applications/{application_id}/policies?version=p3
```

Returns an object of all policies in this particular workspace as shown below

```
{
  absolute_policies: [ ... ],
  default_policies: [ ... ],
  catch_all_action:
}
```

Sample Python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/policies' % application_id, params={'version': '1'})
```

Get Default Policies

This endpoint returns a list of Default policies for a given workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/default_policies
```

Parameters:

Name	Type	Description
id	string	Unique identifier for the policy.
version	string	Indicates the version of the workspace for which to get the policies.
limit	integer	Limits the number of policies per request.
offset	integer	(optional) Offset number received from previous response, should always be used along with `limit`.

Name	Type	Description
consumer_filter_id	string	(optional) Filters the output by the consumer filter id.
provider_filter_id	string	(optional) Filters the output by the provider filter id.

Returns a list of default policies for the provided version of this workspace. The response contains the requested number of polices and an `offset`, to get the next set policies use this `offset` in the subsequent requests. Absence of an `offset` in the response indicates that all the policies are already retrieved.

Sample Python code

```
  application_id = '5f88c996755f023f3bafe163'
  restclient.get('/applications/%s/default_policies' % application_id, params={'version':
'1', 'limit': 3, 'offset': 3})
```

Sample response

```
{
  "results": [
  PolicyObject4,
  PolicyObject5,
  PolicyObject6
  ],
  "offset": 6
}
```

Get Absolute Policies

This endpoint returns a list of Absolute policies in a given workspace. This API is available to API keys with `app_policy_management` capability.

`GET /openapi/v1/applications/{application_id}/absolute_policies`

Parameters:

Name	Type	Description
version	string	Indicates the version of the workspace from which to get the policies.
limit	integer	Limits the number of policies per request.
offset	integer	(optional) Offset number received from previous response, should always be used along with `limit`.
consumer_filter_id	string	(optional) Filters the output by the consumer filter id.
provider_filter_id	string	(optional) Filters the output by the provider filter id.

Returns a list of absolute policies in the provided version of this workspace. The response contains the requested number of polices and an `offset`, to get the next set policies use this `offset` in the subsequent requests. Absence of an `offset` in the response indicates that all the policies are already retrieved.

Sample Python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/absolute_policies' % application_id, params={'version':
'1', 'limit': 3})
```

Sample response

```
{
  "results": [
  PolicyObject1,
  PolicyObject2,
  PolicyObject3
  ],
  "offset": 3
}
```

Get Catch All Policies

This endpoint returns a Catch All policy for a given workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/catch_all
```

Parameters:

Name	Type	Description
version	string	Indicates the version of the workspace from which to get the policies.

Returns a single policy object representing the catch all policy of the given version of the workspace.

Sample Python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/catch_all' % application_id, params={'version': '1'})
```

Get Specific Policy

This endpoint returns an instance of a policy.

```
GET /openapi/v1/policies/{policy_id}
```

Returns the policy object associated with the specified ID.

Sample Python code

```
policy_id = '5f88ca1e755f0222f85ce85c'
restclient.get('/policies/%s' % policy_id)
```

Search for a Specific Policy With Policy Identifier

This endpoint searches for the specified policy using Policy Identifier Parameters as a composite key.

```
POST /openapi/v1/policies/search
```

The query body consists of a JSON body with the following schema:

Name	Type	Description
application_id	string	The ID of the application workspace.
policy_identifier	object	Fields that make up the consistent policy identifier.

The policy identifier fields are made up using the following schema:

Name	Type	Description
version	string	(optional) Indicates the version of the application for which to get the policies; defaults to the latest 'v' version of the application when left unspecified.
consumer_consistent_uuid	string	Consistent UUID of the consumer or source.
provider_consistent_uuid	string	Consistent UUID of the provider or destination.
rank	string	Policy rank has to be one of "DEFAULT" or "ABSOUTE".
action	string	Policy action has to be one of "ALLOW" or "DENY".
priority	integer	Priority value for the policy.
protocol	integer	IP protocol number (0-255) for the .
start_port	integer	(optional) Start of port range (0-65535); defaults to 0 when unspecified.
end_port	integer	(optional) End of port range (0-65535); defaults to 65535 if start_port is 0 or else to the start_prot.

Sample Python code

```
application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
rank = 'DEFAULT'
action = 'ALLOW'
priority = 100
protocol = 6
start_port = 80
version = 'p3'
```

```
  req_body = f'''
  {{
      "application_id": "{application_id}",
      "policy_identifier": {{
          "consumer_consistent_uuid": "{consumer_id}",
          "provider_consistent_uuid": "{provider_id}",
          "rank": "{rank}",
          "action": "{action}",
          "priority": {priority},
          "protocol": "{protocol}",
          "start_port": "{start_port}",
          "version": "{version}"
      }}
  }}'''
  restclient.post('/policies/search', json_body=req_body)
```

Create a Policy

This endpoint is used to create new policies.

```
POST /openapi/v1/applications/{application_id}/policies
```

Parameters:

Attribute	Type	Description
consumer_filter_id	string	ID of a defined filter.
provider_filter_id	string	ID of a defined filter.
version	string	Indicates the version of the workspace in which to update the policies.
rank	string	values can be DEFAULT, ABSOLUTE or CATCHALL for ranking
policy_action	string	values can be ALLOW or DENY: means whether we should allow or drop traffic from consumer to provider on the given service port/protocol
priority	integer	Used to sort policy.

Sample Python code

```
  req_payload = {
    "version": "v1",
    "rank" : "DEFAULT",
    "policy_action" : "ALLOW",
    "priority" : 100,
    "consumer_filter_id" : "123456789",
    "provider_filter_id" : "987654321",
  }
  resp = restclient.post('/openapi/v1/applications/{application_id}/policies',
json_body=json.dumps(req_payload))
```

Create a Default Policy

This endpoint is used to create new default policies. This endpoint creates a default policy similar to the create a policy endpoint.

```
POST /openapi/v1/applications/{application_id}/default_policies
```

Create an Absolute Policy

This endpoint is used to create new absolute policies. This endpoint creates a absolute policy similar to the create a policy endpoint.

```
POST /openapi/v1/applications/{application_id}/absolute_policies
```

Update a Policy

This endpoint updates a policy.

```
PUT /openapi/v1/policies/{policy_id}
```

Parameters:

Attribute	Type	Description
consumer_filter_id	string	ID of a defined filter.
provider_filter_id	string	ID of a defined filter.
policy_action	string	Possible values can be ALLOW or DENY. Indicates whether traffic should be al- lowed or dropped for the given service port/protocol between the consumer and provider.
priority	integer	Used to sort policy priorities

Returns the modified policy object associated with specified ID.

Update a Catch All

This endpoint updates Catch All for a particular workspace.

```
PUT /openapi/v1/applications/{application_id}/catch_all
```

Parameters:

Attribute	Type	Description
version	string	Indicates the version of the workspace in which to update the policies.

Attribute	Type	Description
policy_action	string	Possible values can be ALLOW or DENY. Indicates whether traffic not matching any of the policies in this workspace will be allowed or dropped.

Adding Service Ports to a Policy

This endpoint is used to create service ports for a specific policy.

```
POST /openapi/v1/policies/{policy_id}/l4_params
```

Parameters:

Attribute	Type	Description
version	string	Indicates the version of the workspace from which to get the policies.
start_port	integer	Start port of the range.
end_port	integer	End port of the range.
proto	integer	Protocol Integer value (NULL means all protocols).
description	string	(optional) Short string about this proto and port.

Updating Service Ports of a Policy

This endpoint updates the specified service port of a Policy.

```
PUT /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Parameters:

Attribute	Type	Description
approved	bool	Marks the policy as approved.

Deleting Service Ports of a Policy

This endpoint deletes the specified service port of a policy. (Optional) See Exclusion Filters for more details.

```
DELETE /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Parameters:

Attribute	Type	Description
create_exclusion_filter	bool	(Optional) If true, creates an exclusion filter matching the policy. Flows matching this filter will be excluded from future automatic policy discovery runs. See Exclusion Filters for more details.

Deleting a Policy

This endpoint deletes the specified Policy. No exclusion filters are created.

```
DELETE /openapi/v1/policies/{policy_id}
```

Deleting a Policy with Identifier

This endpoint deletes the specified policy using Policy Identifier Parameters. No exclusion filters are created.

```
DELETE /openapi/v1/policies/destroy_with_identifier
```

The query body consists of a JSON body with the following schema:

Name	Type	Description
application_id	string	The ID of the application workspace.
policy_identifier	object	Fields that make up the consistent policy identifier.

The policy identifier fields are made up using the following schema:

Name	Type	Description
version	string	(optional) 'v' version of the application workspace in which to perform the delete operation; defaults to the latest 'v' version of the workspace when unspecified.
consumer_consistent_uuid	string	Consistent UUID of the consumer or source
provider_consistent_uuid	string	Consistent UUID of the provider or destination
rank	string	Policy rank has to be one of “DEFAULT” or “ABSOUTE”
action	string	Policy action has to be one of “ALLOW” or “DENY”
priority	integer	Priority value for the policy

Name	Type	Description
protocol	integer	IP protocol number (0-255) for the policy
start_port	integer	(optional) Start of port range (0-65535); defaults to 0 when unspecified
end_port	integer	(optional) End of port range (0-65535); defaults to 65535 if start_port is 0 or else to the start_prot

Sample Python code

```
application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
action = 'ALLOW'
rank = 'DEFAULT'
protocol = 6
start_port = 80
priority = 100
version = '5'

req_body = f'''
{{
    "application_id": "{application_id}",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{consumer_id}",
        "provider_consistent_uuid": "{provider_id}",
        "rank": "{rank}",
        "priority": {priority},
        "action": "{action}",
        "protocol": "{protocol}",
        "start_port": "{start_port}",
        "version": "{version}"
    }}
}}'''
restclient.delete('/policies/destroy_with_identifier', json_body=req_body)
```

Policy Quick Analysis

This endpoint can be used to find matching set of policies for any hypothetical flow against the analyzed/enforced polices in a root scope. For more details refer Quick Analysis

This API is only available to users with a minimum read access to root scope and requires `app_policy_management` capability associated with the API key.

`POST /openapi/v1/policies/{rootScopeID}/quick_analysis`

The query body consists of a JSON body with the following schema:

Name	Type	Description
consumer_ip	string	IP Address of the client / consumer.

Name	Type	Description
provider_ip	string	IP Address of the server / provider.
provider_port	integer	(optional) Provider Port, only relevant for TCP or UDP flows.
protocol	string	Protocol of the flow, e.g. TCP.
analysis_type	string	Analysis type can be either **analyzed** or **enforced**. Analysis type “analyzed” makes the flow decision by matching the flow against all the analyzed polices in the root scope. Analysis type “enforced” makes the flow decision by matching the flow against all enforced policies in the root scope.
application_id	string	(optional) The ID of the primary workspace, always accompanied by the workspace ‘v’ version, if specified, makes the flow decision by using the policies from the specified version along with analyzed/enforced policies from other workspaces in the root scope. If this field is skipped, the flow decision is made by considering all the analyzed/enforced polices in the root scope.
version	integer	(optional) The ‘v’ version of the workspace mentioned above. This must be specified if the application_id is specified and must be skipped otherwise.

Sample request

The body of the request should be a JSON formatted query.

An example of a query body where the flow decision is based on all analyzed polices:

```
    req_payload = {
        "consumer_ip": "4.4.1.1",
        "provider_ip": "4.4.2.1",
        "provider_port": 9081,
        "protocol": "TCP",
        "analysis_type": "analyzed"
    }
    resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis',
json_body=json.dumps(req_payload))
```

An example of a query body where the flow decision is based on the policies from the workspace's ‘v’ version along with the analyzed polices from all other workspaces in the root scope:

```
    req_payload = {
        "consumer_ip": "4.4.1.1",
        "provider_ip": "4.4.2.1",
        "provider_port": 9081,
        "protocol": "TCP",
        "analysis_type": "analyzed",
        "application_id": "5e7e5f56497d4f0bc26c7bb3",
        "version": 1
    }
    resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis',
json_body=json.dumps(req_payload))
```

Sample response

The response is a JSON object in the body with the following properties:

Keys	Values
policy_decision	The decision of the hypothetical flow whether is allowed or denied.
outbound_policy	The policy on the consumer thats allowing/denying the outgoing traffic
inbound_policy	The policy on the provider thats allowing/denying the incoming traffic

```
    {
        "policy_decision": "ALLOW",
        "outbound_policy": {
            "policy_rank": "DEFAULT",
            "start_port": 9082,
            "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
            "src_filter_id": "5e7e600e497d4f7341f4f459",
            "end_port": 9082,
            "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
            "dst_filter_id": "5e7d0efc497d4f44b6b09351",
            "action": "ALLOW",
            "protocol": "TCP",
            "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
        },
        "inbound_policy": {
            "policy_rank": "DEFAULT",
            "start_port": 9082,
            "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
            "src_filter_id": "5e7e600e497d4f7341f4f459",
            "end_port": 9082,
            "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
            "dst_filter_id": "5e7d0efc497d4f44b6b09351",
            "action": "ALLOW",
            "protocol": "TCP",
            "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
        }
    }
```

Policy Statistics

This endpoint returns the number of packets, bytes, and conversations observed for a policy over a time interval. A conversation can be broadly described as a flow observation matching a policy that is aggregated with a granularity of one hour. The number of conversations that are measured for a given policy within one hour represents the number of distinct pairs of consumer and provider inventory items that have communicated over the network during that one hour.

Although this endpoint accepts Policy Identifier Parameters as input, we recommend you to use policy and L4 parameter IDs from a published version of the workspace.

Note After a new version of the application workspace is published, it can take up to 6 hours before results become available. All the timestamp resolutions will also have a minimum granularity of 6 hours.

To get the policy statistics for a policy across enforced versions of an application workspace the URL path is:

```
POST /openapi/v1/policies/stats/enforced
```

To get the policy statistics for a policy across analyzed versions of an application workspace the URL path is:

```
POST /openapi/v1/policies/stats/analyzed
```

The query body consists of a JSON body with the following schema:

Table 4:

Name	Type	Description
application_id	string	The ID of the application workspace.
t0	string	The beginning of the time interval in RFC-3339 format.
t1	string	(optional) The end of the time interval in RFC-3339 format; defaults to current time if left unspecified.
policy_id	string	The ID of the policy; not required if the policy identifier is present.
l4_param_id	string	The ID of the l4 parameter; not required if the policy identifier is present, or for “CATCH_ALL” policies.
policy_identifier	object	Fields that make up the consistent policy identifier.

The policy identifier fields are made up using the following schema:

Name	Type	Description
consumer_consistent_uuid	string	Consistent UUID of the consumer or source.
provider_consistent_uuid	string	Consistent UUID of the provider or destination.
rank	string	Policy rank has to be one of “DEFAULT” or “ABSOUTE”.
action	string	Policy action has to be one of “ALLOW” or “DENY”.
priority	integer	Priority value for the policy.
protocol	integer	IP protocol number (0-255) for the policy.
start_port	integer	(optional) Start of port range (0-65535); defaults to 0 when unspecified
end_port	integer	(optional) End of port range (0-65535); defaults to 65535 if start_port is 0 or else to the start_prot.

Sample Python code

```
application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
action = 'ALLOW'
rank = 'DEFAULT'
protocol = 6
start_port = 80
priority = 100

req_body = f'''
{{
    "application_id": "{application_id}",
    "t0":"2022-07-06T00:00:00Z",
    "t1":"2022-07-28T19:00:00Z",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{consumer_id}",
        "provider_consistent_uuid": "{provider_id}",
        "rank": "{rank}",
        "priority": {priority},
        "action": "{action}",
        "protocol": "{protocol}",
        "start_port": "{start_port}"
    }}
}}'''
restclient.post('/policies/stats/analyzed', json_body=req_body)

# For CATCH_ALL policies:
root_app_scope_id = '6f88ca1e755f0222f85ce85e'
```

```
rank = 'CATCH_ALL'
action = 'DENY'
req_body = f'''
{{
    "application_id": "{application_id}",
    "t0":"2022-07-06T00:00:00Z",
    "t1":"2022-07-28T19:00:00Z",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{root_app_scope_id}",
        "provider_consistent_uuid": "{root_app_scope_id}",
        "rank": "{rank}",
        "action": "{action}"
    }}
}}'''

restclient.post('/policies/stats/analyzed', json_body=req_body)
```

Sample response

The response is a JSON object in the body with the following properties.

Table 5:

Keys	Values
conversation_count	The number of conversations that are observed for the specified duration and policy.
packet_count	The number of packets that are observed for the specified duration and policy.
byte_count	The number of bytes observed for the specified duration and policy.
first_seen_at	The timestamp (in RFC-3339 format) when we first observed flows for this policy.
last_seen_at	The timestamp (in RFC-3339 format) when we last observed flows for this policy.
agg_start_version	The earliest published version of this policy on record from time t0 onwards.
agg_start_time	The timestamp the agg_start_version was published.

```
{
  "conversation_count": 72,
  "packet_count": 800,
  "byte_count": 1960,
  "first_seen_at": "2022-09-09T11:00:00.000Z",
  "last_seen_at": "2022-09-09T11:00:00.000Z",
  "agg_start_version": 4,
  "agg_start_time": "2022-08-10T23:00:00.000Z"
}
```

Unused Policies

This endpoint returns the policy identifiers in an published workspace for which no conversations are observed over a specified time interval.

Policy Identifier

All policies and ADM-generated clusters can change their IDs across application workspace versions even if the underlying filter queries or policy port and protocol do not change. In order to keep track of flow hit counts for a particular policy across workspace versions we use consistent UUIDs for the filters that do not change across versions and a composite key called the *policy identifier* comprising the provider and consumer consistent UUIDs along with rank, action, priority, port and protocol.

Thus, policy identifiers serve as a composite key that can both identify and describe the important aspects of a policy across application workspace versions, whereas policy IDs (such as those used in the regular CRUD endpoints) can change across versions of the workspace.

Note Provider/Consumer consistent UUIDs and policy identifiers do not uniquely identify a filter or policy as they are shared across different application workspace versions.

To perform CRUD operations on a particular cluster or policy it is recommended to resolve the identifier to a concrete policy for a specific application workspace version the search endpoint.

Regular CRUD operations can be performed using policy IDs whereas only Policy Statistics and Destroy With Identifier accept the policy identifier as input. This is mainly for convenience to avoid the intermediate call to search, and instead directly validate and destroy all unused policies in a workspace.

It is strongly recommended that policy and filter IDs are used wherever possible and to not manually generate policy identifiers for the Policy Statistics or Destroy With Identifier API endpoints. However, the following example illustrates one way of generating policy identifiers from the policy object:

```
resp = restclient.get(f'/policies/631b0590497d4f09b537b973')
policy = resp.json() # policy object
policy_identifer = {
    'consumer_consistent_uuid': policy['consumer_filter']['consistent_uuid'],
    'provider_consistent_uuid': policy['provider_filter']['consistent_uuid'],
    'rank': policy['rank'],
    'action': policy['action'],
    'priority': policy['priority'],
    'protocol': policy['l4_params'][0]['proto'],
    'start_port': policy['l4_params'][0]['port'][0],
    'end_port': policy['l4_params'][0]['port'][1]
}
```

Note After a new version of the application workspace is published it can take up to 6 hours before results become available. All the timestamp resolutions will also have a minimum granularity of 6 hours.

To get the unused policies across enforced versions of an application workspace the URL path is:

```
POST /openapi/v1/unused_policies/{application_id}/enforced
```

To get the unused policies across analyzed versions of an application workspace the URL path is:

```
POST /openapi/v1/unused_policies/{application_id}/analyzed
```

The query body consists of a JSON body with the following schema:

Name	Type	Description
t0	string	The beginning of the time interval in RFC-3339 format.
t1	string	(Optional) The end of the time interval in RFC-3339 format; defaults to current time if left unspecified.
limit	integer	(Optional) Limits the number of policies per request.
offset	string	(Optional) Offset received from previous response – useful for pagination.

```
application_id = '62e1915e755f026f2bcdd805'
resp = restclient.post(f'/unused_policies/{application_id}/analyzed', json_body=f'''
{{
    "t0":"2022-07-06T00:00:00Z",
    "t1":"2022-07-28T19:00:00Z"
}}''')'
```

Sample response

The response is a JSON object in the body with the following properties.

Keys	Values
application_id	The ID of the application workspace.
policy_identifiers	A list of policy identifiers of the unused policies.
offset	Response offset to be passed for the next page of results.

To generate the next page of results, take the object received by the response in offset and pass it as the value for the offset of the next query.


```
{
  "application_id": "63054a97497d4f2dc113a9c4",
  "policy_identifiers": [
    {
      "consumer_consistent_uuid": "62fff45c497d4f5064973c4d",
      "provider_consistent_uuid": "62fff45c497d4f5064973c4d",
      "version": "p1",
      "rank": "DEFAULT",
      "policy_action": "ALLOW",
      "priority": 10,
      "proto": 6,
      "start_port": 10000,
      "end_port": 10000,
      "agg_start_version": 1,
      "agg_start_time": "2022-08-10T23:00:00.000Z"
```

```
    },
    {
      "consumer_consistent_uuid": "62fff45c497d4f5064973c4d",
      "provider_consistent_uuid": "62fff45c497d4f5064973c4d",
      "version": "p1",
      "rank": "DEFAULT",
      "policy_action": "ALLOW",
      "priority": 10,
      "protocol": 6,
      "start_port": 10001,
      "end_port": 10001,
      "agg_start_version": 1,
      "agg_start_time": "2022-08-10T23:00:00.000Z"
    }
  ],
  "offset": "eyJvZmZzZXQiOjZ9"
}
```


Policy Templates

This set of APIs can be used to add, edit or delete Policy Templates and require the app_policy_management capability associated with the API key.

Get Policy Templates

This endpoint returns a list of policy templates for a particular root scope. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/application_templates?root_app_scope_id={root_app_scope_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_app_scope_id	string	The unique identifier of the root scope.

Response object: Returns a list of policy template objects for the specified root scope.

Sample python code

```
root_app_scope_id = '<root-app-scope-id>'
restclient.get('/application_templates?root_app_scope_id=%s' % root_app_scope_id)
```

Get Specific Policy Template

This endpoint returns an instance of policy templates.

```
GET /openapi/v1/application_templates/{template_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
template_id	string	The unique identifier for the policy template.

Response object: Returns the policy template object with the specified ID.

Sample python code

```
template_id = '<template-id>'
restclient.get('/application_templates/%s' % template_id)
```

Create a Policy Template

This endpoint is used to create a new policy template.

```
POST /openapi/v1/application_templates
```

The JSON request body contains the following keys

Attribute	Type	Description
name	string	Used as the name of the template during import.
description	string	(optional) Template description displayed during the apply process
parameters	parameters object	Template parameters, see below.
absolute_policies	array of policy objects	(optional) Array of absolute policies.
default_policies	array of policy objects	(required) Array of default policies, can be empty.

Response object: Returns the created policy template object.

Sample python code

```
root_app_scope_id = '<root-app-scope-id>'
payload = {'root_app_scope_id': root_app_scope_id,
           'name': "policy_name",
           'default_policies': [
              {
                'action': 'ALLOW',
                'priority': 100,
                'l4_params': [
                  {
                    'proto': 17,
                    'port': [80, 90]
                  }
                ]
              }
            ]
          }
restclient.post('/application_templates',
                json_body=json.dumps(payload))
```

Update a Policy Template

This endpoint updates a policy template.

```
PUT /openapi/v1/application_templates/{template_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
template_id	string	The unique identifier for the policy template.

The JSON request body contains the following keys

Attribute	Type	Description
name	string	(optional) Used as the name of the template during import.
description	string	(optional) Template description displayed during the apply process

Response object: Returns the modified policy template object with the specified ID.

Sample python code

```
new_name = <new-name>
payload = {'name': new_name}
template_id = '<template-id>'
restclient.post('/application_templates/%s' % template_id,
                json_body=json.dumps(payload))
```

Deleting a Policy Template

This endpoint deletes the specified policy template.

```
DELETE /openapi/v1/application_templates/{template_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
template_id	string	The unique identifier for the policy template.

Response object: None

Sample python code

```
template_id = '<template-id>'
restclient.delete('/application_templates/%s' % template_id)
```

Download a Policy Template

This endpoint downloads a policy template.

```
GET /openapi/v1/application_templates/{template_id}/download
```

Parameters: The request URL contains the following parameters

Name	Type	Description
template_id	string	The unique identifier for the policy template.

Response object: Returns the full policy template definition with the specified ID.

Sample python code

```
template_id = '<template-id>'
restclient.get('/application_templates/%s/download' % template_id)
```

Clusters

This set of APIs can be used to add, edit or delete Clusters, which are members of workspaces (“applications”). They require the `user_role_scope_management` capability associated with the API key.

Cluster object

The cluster object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the cluster.
consistent_uuid	string	An id that is consistent across automatic policy discovery runs.
application_id	string	The id for the workspace to which the cluster belongs.
version	string	The version of the workspace to which the cluster belongs
name	string	The name of the cluster.
description	string	The description of the cluster.
approved	boolean	If the cluster has been ‘approved’ by the user.
query	JSON	Filter (or match criteria) associated with the filter in conjunction with the filters of the parent scopes.
short_query	JSON	Filter (or match criteria) associated with the filter.
alternate_queries	array of queries	Alternate suggested queries generated by an automatic policy discovery run in dynamic mode.

Attribute	Type	Description
inventory	array of inventory	If requested, returns member inventory of the cluster including IP, hostname, vrf_id and uuid.

Get Clusters

This endpoint returns a list of clusters for a particular workspace (“application”). This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/clusters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The id for the workspace to which the cluster belongs.
version	string	Indications the version of the workspace for which to get the clusters.
include_inventory	boolean	Include the inventory of the clusters.

Response object: Returns an array of all clusters for this particular workspace and version.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s/clusters' % application_id)
```

Get Specific Cluster

This endpoint returns an instance of a cluster.

```
GET /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
cluster_id	string	Unique identifier for the cluster.
include_inventory	boolean	Include the inventory of the clusters.

Response object: Returns the cluster object associated for the specified ID.

Sample python code

```
cluster_id = '5d02d021497d4f0949ba74e4'
```

```
restclient.get('/clusters/%s' % cluster_id)
```

Create a Cluster

This endpoint is used to create a new cluster.

```
POST /openapi/v1/applications/{application_id}/clusters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The id for the workspace to which the cluster belongs.

The JSON query body contains the following keys

Attribute	Type	Description
name	string	The name of the cluster.
version	string	Indicates the version of the workspace the cluster will be added to.
description	string	(optional) The description of the cluster.
approved	boolean	(optional) An approved cluster will not be updated during an automatic policy discovery run. Default false.
query	JSON	Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the workspace, otherwise ignored.
query	JSON	Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the workspace, otherwise ignored.
nodes	Array	List of ip addresses or endpoints. Will be used to create the query matching these ips unless a query is provided and the workspace is in Dynamic Mode.

Nodes object attributes:

Name	Type	Description
ip	string	IP address
name	string	(optional) The name of the node.
prefix_len	integer	(optional) Subnet mask.

Note The nodes will be used to create a query unless a query is provided and the workspace is in Dynamic Mode.

Response object: Returns the newly created cluster object.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
payload = {
  'name': 'test_cluster',
  'version': 'v2',
  'description': 'basic granularity',
  'approved': False,
  'query': {
      'type': 'eq',
      'field': 'host_name',
      'value': 'centos6001'
  }
}
restclient.post('/applications/%s/clusters' % application_id)
```

Update a Cluster

This endpoint updates a cluster.

```
PUT /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
cluster_id	string	Unique identifier for the cluster.

The JSON query body contains the following keys

Attribute	Type	Description
name	string	The name of the cluster.
description	string	(optional) The description of the cluster.
approved	boolean	An approved cluster will not be updated during an automatic policy discovery run.

Attribute	Type	Description
query	JSON	Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the workspace, otherwise ignored.

Response object: Returns the modified cluster object associated with specified ID.

Sample python code

```
cluster_id = '5d02d2a4497d4f5194f104ef'
payload = {
  'name': 'new_test_cluster',
}
restclient.put('/clusters/%s' % cluster_id, json_body=json.dumps(payload))
```

Deleting a Cluster

This endpoint deletes the specified Cluster. If the cluster is used by any policies the cluster will not be deleted and a list of dependents will be returned.

```
DELETE /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
cluster_id	string	Unique identifier for the cluster.

Response object: None

Sample python code

```
cluster_id = '5d02d2a4497d4f5194f104ef'
restclient.delete('/clusters/%s' % cluster_id)
```

Conversations

Conversations are aggregated flows in the time range of an automatic policy discovery run where the consumer port is removed. More detailed description about the conversations can be found in Conversations.

This API enables you to search the conversations generated during an automatic policy discovery run for a given workspace. It requires `app_policy_management` capability associated with the API key to invoke this API.

Search Conversations in a Policy Discovery Run

This end point enables you to search the conversations in an automatic policy discovery run for a given workspace. You can also specify a subset of supported dimensions and metrics which you may want to see as part of the downloaded conversations. Optionally, you can query for a subset of conversations using filters on supported dimensions and metrics.

```
POST /openapi/v1/conversations/{application_id}
```

The query consists of a JSON body with the following keys.

Name	Type	Description
version	integer	Version of the automatic policy discovery run
filter	JSON	(optional) Query filter. If filter is empty (i.e. {}), then query matches all the conversations. More specific conversations can be downloaded using filters on supported dimensions and metrics. For the syntax on filters refer to Filters .
dimensions	array	(optional) List of dimensions to be returned for the downloaded conversations. The list of supported dimension can be found Supported Dimensions .
metrics	array	(optional) List of metrics to be returned for the downloaded conversations. The list of supported metrics can be found Supported metrics .
limit	integer	(optional) Number of conversations to be returned in a single API response.
offset	string	(optional) Offset received from previous response – useful for pagination.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "version": 1,
    "filter": {
        "type": "and",
        "filters":[
            {
                "type": "eq",
                "field": "excluded",
                "value": False
            },
            {
                "type": "eq",
                "field": "protocol",
                "value": "TCP"
            },
        ]
    },
    "dimensions": ["src_ip", "dst_ip", "port"],
```

```
        "metrics": ["byte_count", "packet_count"],
        "limit" : 2,
        "offset": <offset-object>
    }
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
offset	Response offset to be passed for the next page of results
results	List of results

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

```
   req_payload = {"version": 1,
                  "limit": 10,
                  "filter": {"type": "and",
                      "filters": [
                          {"type": "eq", "field": "excluded", "value": False},
                          {"type": "eq", "field": "protocol", "value": "TCP"}
                       ]
                  }
    }

    resp = restclient.post('/conversations/{application_id}',
json_body=json.dumps(req_payload))
    print resp.status_code
    if resp.status_code == 200:
        parsed_resp = json.loads(resp.content)
        print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Top N Conversations in a Policy Discovery Run

This endpoint enables you to search the top conversations for an automatic policy discovery that is run for a given workspace based on a metric and grouped by a dimension. The current supported metrics are Supported metrics and the current supported group by dimensions are Supported Dimensions you can query for a subset of conversations using filters on supported dimensions and metrics. For example, you can search for the source IP address with the most byte traffic conversations using a query with the `src_ip` dimension with the `byte_count` metric.

`POST /openapi/v1/conversations/{application_id}/topn`

The query consists of a JSON body with the following keys.

Name	Type	Description
version	integer	Version of the automatic policy discovery run

Name	Type	Description
dimension	string	The dimension for the conversations to be grouped by for the top N query. Supported dimensions: src_ip, dst_ip
metric	string	The metric to be sorted by for the top N conversations. The list of supported metrics can be found Supported metrics .
filter	JSON	(optional) Query filter. If filter is empty (i.e. {}), then query matches all the conversations. More specific conversations can be downloaded using filters on supported dimensions and metrics. For the syntax on filters, see Filters .
threshold	integer	Number of top N results to be returned in a single API response.

The body of the request should be a JSON-formatted query. An example of a query body is shown below.

```
{
    "version": 1,
    "dimension": "src_ip",
    "metric": "byte_count",
    "filter": {
        "type": "and",
        "filters":[
            {
                "type": "eq",
                "field": "excluded",
                "value": False
            },
            {
                "type": "eq",
                "field": "protocol",
                "value": "TCP"
            },
        ]
    },
    "threshold" : 10
}
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
results	List with one JSON object with a results key and a value of a list of results objects with keys matching the query dimension and metric.

```
[ {"result": [
  {
    "byte_count": 1795195565,
    "src_ip": "192.168.1.6"
  },
  {
    "byte_count": 1781002379,
    "src_ip": "192.168.1.28"
  },
  ...
] } ]
```

```
  req_payload = {"version": 1, "dimension": "src_ip", "metric": "byte_count",
    "filter": {"type": "and",
      "filters": [
        {"type": "eq", "field": "excluded", "value": False},
        {"type": "eq", "field": "protocol", "value": "TCP"},
       {"type": "eq", "field": "consumer_filter_id", "value": "16b12a5614c5af5b68afa7ce"},

        {"type": "subnet", "field": "src_ip", "value": "192.168.1.0/24"}
      ]
    },
    "threshold" : 10
  }

  resp = restclient.post('/conversations/{application_id}/topn',
json_body=json.dumps(req_payload))
  print resp.status_code
  if resp.status_code == 200:
      parsed_resp = json.loads(resp.content)
      print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Supported Dimensions

Name	Type	Description
src_ip	string	IP address of the consumer
dst_ip	string	IP address of the provider
protocol	string	Protocol used in the communication. Ex: “TCP”, “UDP” and so on
port	integer	Port of the provider.
address_type	string	“IPv4” or “IPv6”

Name	Type	Description
consumer_filter_id	string	Cluster ID of the cluster if the consumer IP belongs to a cluster, else the Scope ID the consumer IP belongs to.
provider_filter_id	string	Cluster ID of the cluster if the provider IP belongs to a cluster, else the Scope ID the provider IP belongs to.
excluded	boolean	Whether this conversation is excluded while generating policies.
confidence	double	The confidence level of consumer and provider classification. The value varies from 0.0 to 1.0 with 1.0 being more confident about classification.

Supported metrics

Name	Type	Description
byte_count	integer	Total number of bytes in the conversation
packet_count	integer	Total number of packets in the conversation

Exclusion Filters

This set of APIs can be used to add, edit or delete Exclusion Filters and require the `user_role_scope_management` capability associated with the API key.

Exclusion Filters exclude flows from the automatic policy discovery clustering algorithm. See Exclusion Filters for more information.

Exclusion Filter object

The exclusion filter object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the cluster.
application_id	string	The id for the workspace to which the exclusion filter belongs.
version	string	The version of the workspace to which the exclusion filter belongs.

Attribute	Type	Description
consumer_filter_id	string	ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy.
proto	integer	Protocol Integer value (NULL means all protocols).
port	array	Inclusive range of ports. eg [80, 80] or [5000, 6000]. NULL means all ports.
updated_at	integer	Unix timestamp of when the exclusion filter was updated.

Get Exclusion Filters

This endpoint returns a list of exclusion filters for a particular workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/exclusion_filters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.
version	string	Indicates the version of the workspace for which to get the exclusion filters.

Response object: Returns a list of exclusion filter objects for the specified workspace and version.

Sample python code

```
application_id = '<application-id>'
params = {'version': 'v10'}
restclient.get('/applications/%s/exclusion_filters' % application_id,
               params=params)
```

Get Specific Exclusion Filter

This endpoint returns an instance of an exclusion filters.

```
GET /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
exclusion_filter_id	string	The unique identifier for the exclusion filter.

Response object: Returns the exclusion filter object with the specified ID.

Sample python code

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.get('/exclusion_filters/%s' % exclusion_filter_id)
```

Create an Exclusion Filter

This endpoint is used to create a new exclusion filter.

```
POST /openapi/v1/applications/{application_id}/exclusion_filters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the workspace.

The JSON request body contains the following keys

Attribute	Type	Description
version	string	The version of the workspace to which the exclusion filter belongs.
consumer_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy.
proto	integer	(optional) Protocol Integer value (NULL means all protocols).
start_port	integer	(optional) Start port of the range.
end_port	integer	(optional) End port of the range.

Missing optional parameters will be considered as wildcards (match any).

Response object: Returns the created exclusion filter object.

Sample python code

```
provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'version': 'v0',
           'consumer_filter_id': consumer_filter_id,
```

```
            'provider_filter_id': provider_filter_id,
            'proto': 6,
            'start_port': 800,
            'end_port': 1000}
application_id = '<application-id>'
restclient.post('/applications/%s/exclusion_filters' % application_id,
                json_body=json.dumps(payload))
```

Update an Exclusion Filter

This endpoint updates an exclusion filter.

```
PUT /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
exclusion_filter_id	string	The unique identifier for the exclusion filter.

The JSON request body contains the following keys

Attribute	Type	Description
consumer_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy.
proto	integer	Protocol Integer value (NULL means all protocols).
start_port	integer	(optional) Start port of the range.
end_port	integer	(optional) End port of the range.

Response object: Returns the modified exclusion filter object with the specified ID.

Sample python code

```
payload = {'proto': 17}
exclusion_filter_id = '<exclusion-filter-id>'
restclient.post('/exclusion_filters/%s' % exclusion_filter_id,
                json_body=json.dumps(payload))
```

Deleting an Exclusion Filter

This endpoint deletes the specified exclusion filter.

```
DELETE /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
exclusion_filter_id	string	The unique identifier for the exclusion filter.

Response object: None

Sample python code

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.delete('/exclusion_filters/%s' % exclusion_filter_id)
```

Default Exclusion Filters

This set of APIs can be used to add, edit or delete Default Exclusion Filters and require the `app_policy_management` capability associated with the API key.

Exclusion Filters exclude flows from the automatic policy discovery clustering algorithm. See Exclusion Filters for more information.

Default Exclusion Filter object

The exclusion filter object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the default exclusion filter.
consumer_filter_id	string	ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy.
proto	integer	Protocol Integer value (NULL means all protocols).
port	array	Inclusive range of ports. eg [80, 80] or [5000, 6000]. NULL means all ports.
updated_at	integer	Unix timestamp of when the exclusion filter was updated.

Get Default Exclusion Filters

This endpoint returns a list of default exclusion filters. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/default_exclusion_filters?root_app_scope_id={root_app_scope_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_app_scope_id	string	The unique identifier of the root scope.

Response object: Returns a list of default exclusion filter objects for the root scope.

Sample python code

```
root_app_scope_id = '<root-app-scope-id>'
 restclient.get('/default_exclusion_filters?root_app_scope_id=%s' % root_app_scope_id)
```

Get Specific Default Exclusion Filter

This endpoint returns an instance of a default exclusion filters.

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'

restclient.get('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Parameters: The request URL contains the following parameters

Name	Type	Description
default_exclusion_filter_id	string	The unique identifier for the exclusion filter.

Response object: Returns the default exclusion filter object with the specified ID.

Sample python code

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.get('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Create a Default Exclusion Filter

This endpoint is used to create a new default exclusion filter.

```
POST /openapi/v1/default_exclusion_filters?root_app_scope_id={root_app_scope_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_app_scope_id	string	The unique identifier of the root scope.

The JSON request body contains the following keys

Attribute	Type	Description
consumer_filter_id	string	(optional) ID of a defined scope or inventory filter.
provider_filter_id	string	(optional) ID of a defined scope or inventory filter.
proto	integer	(optional) Protocol Integer value (NULL means all protocols).
start_port	integer	(optional) Start port of the range.
end_port	integer	(optional) End port of the range.

Response object: Returns the created default exclusion filter object.

Sample python code

```
provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'consumer_filter_id': consumer_filter_id,
           'provider_filter_id': provider_filter_id,
           'proto': 6,
           'start_port': 800,
           'end_port': 1000}
root_app_scope_id = '<root-app-scope-id>'
restclient.post('/default_exclusion_filters?root_app_scope_id=%s' % root_app_scope_id,
                json_body=json.dumps(payload))
```

Update a Default Exclusion Filter

This endpoint updates a default exclusion filter.

```
PUT /openapi/v1/default_exclusion_filters/{default_exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
default_exclusion_filter_id	string	The unique identifier for the default exclusion filter.

The JSON request body contains the following keys

Attribute	Type	Description
consumer_filter_id	string	(optional) ID of a defined scope or inventory filter.
provider_filter_id	string	(optional) ID of a defined scope or inventory filter.
proto	integer	Protocol Integer value (NULL means all protocols).
start_port	integer	(optional) Start port of the range.

end_port	integer	(optional) End port of the range.

Response object: Returns the modified default exclusion filter object with the specified ID.

Sample python code

```
payload = {'proto': 17}
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.post('/default_exclusion_filters/%s' % default_exclusion_filter_id,
                json_body=json.dumps(payload))
```

Deleting a Default Exclusion Filter

This endpoint deletes the specified default exclusion filter.

```
DELETE /openapi/v1/default_exclusion_filters/{default_exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
default_exclusion_filter_id	string	The unique identifier for the exclusion filter.

Response object: None

Sample python code

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.delete('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Live Analysis

Live analysis or Policy Analysis is an important aspect of generating security policies. It allows you to evaluate the impact of a set of policies – where generated by automatic policy discovery or manually added by users – before actually enforcing those policies on the workloads. Live analysis allows users to run what-if analysis on live traffic without disrupting any application traffic.

The set of APIs available in this section allow downloading flows and the effect of current set of published policies in a workspace on those flows. It requires `app_policy_management` capability associated with the API key to invoke these set of APIs.

Flows available via Live Analysis have some attributes (dimensions and metrics) and the download API allows user to filter flows by different criteria on dimensions.

Flow dimensions available in Live Analysis

This endpoint is useful to know the columns on which search criteria (or *filters*) can be specified for downloading flows available via Live Analysis. Most common use case would be to download *permitted*, *escaped* or *rejected* flows this can be achieved by passing a search criteria on category dimension to the download API. When used with **type: eq** the flow's inbound and outbound category must match. When used with **type: contains** the flows inbound or outbound category must match

```
GET /openapi/v1/live_analysis/dimensions
```

Flow metrics available in Live Analysis

This endpoint returns the list of metrics (e.g. byte count, packet count) associated with live analysis. One use case for this endpoint would be to project a subset of metrics in the download API, i.e. instead of downloading all the metrics, users can specify a small subset of metrics they are interested in.

```
GET /openapi/v1/live_analysis/metrics
```

Download flows available via Live Analysis

This endpoint returns the list of flows matching the filter criteria. Each flow object in the result has attributes that are a union of live analysis dimensions (returned by the live analysis dimensions API above) as well as the live analysis metrics (returned by the live analysis metrics API above). Optionally, user can also specify a small subset of dimensions or metrics if they are not interested in the full set of available dimensions and metrics – this projection of a smaller subset of dimensions or metrics also have the side effect of making API calls fast.

```
POST /openapi/v1/live_analysis/{application_id}
```

The query body consists of a JSON body with the following keys.

Name	Type	Description
t0	integer or string	Start of time interval (epoch or ISO 8601)
t1	integer or string	End of time interval (epoch or ISO 8601)
filter	JSON	Query filter. If filter is empty (i.e. {}), then query matches all flows. Refer to section on Filters in Flow Search regarding syntax of filters.
dimensions	array	(optional) List of flow dimensions to be returned for the downloaded flows available through Live Analysis. If unspecified, all available dimensions are returned.
metrics	array	(optional) List of flow metrics to be returned for the downloaded flows available through Live Analysis.
limit	integer	(optional) Number of flows to be returned in a single API response.
offset	string	(optional) Offset received from previous response – useful for pagination.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "t0": "2016-06-17T09:00:00-0700",
    "t1": "2016-06-17T17:00:00-0700",
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "category",
                "value": "escaped"
            },
            {
                "type": "in",
                "field": "dst_port",
                "values": ["80", "443"]
            }
        ]
    },
    "limit": 100,
    "offset": <offset-object>
}
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
offset	Response offset to be passed for the next page of results
results	List of results

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

Sample python code

```
   req_payload = {"t0": "2016-11-07T09:00:00-0700",
                  "t1": "2016-11-07T19:00:00-0700",
                  "limit": 10,
                  "filter": {"type": "and",
                      "filters": [
                          {"type": "contains", "field": "category", "value": "escaped"},
                          {"type": "regex", "field": "src_hostname", "value": "web*"}
                       ]
                  }
    }

    resp = restclient.post('/live_analysis/{application_id}',
json_body=json.dumps(req_payload))
    print resp.status_code
    if resp.status_code == 200:
        parsed_resp = json.loads(resp.content)
        print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Scopes

This set of APIs can be used to manage Scopes (or AppScopes) in Secure Workload cluster deployment. They require the `user_role_scope_management` capability associated with the API key. The API to get the list of scopes is also available to API keys with `app_policy_management` or `sensor_management` capability.

Scope object

The scope object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the scope.
short_name	string	User specified name of the scope.
name	string	Fully qualified name of the scope. This is a fully qualified name, i.e. it has name of parent scopes (if applicable) all the way to the root scope.
description	string	User specified description of the scope.
short_query	JSON	Filter (or match criteria) associated with the scope.
query	JSON	Filter (or match criteria) associated with the scope in conjunction with the filters of the parent scopes (all the way to the root scope).
vrf_id	integer	ID of the VRF to which scope belongs to.
parent_app_scope_id	string	ID of the parent scope.
child_app_scope_ids	array	An array of scope children's ids.
policy_priority		Used to sort workspace priorities. See Semantics and Viewing.
dirty	bool	Indicates a child or parent query has been updated and that the changes need to be committed.
dirty_short_query	JSON	Non-null if the query for this scope has been updated but not yet committed.

Get scopes

This endpoint returns a list of scopes known to Secure Workload appliance. This API is available to API keys with either `app_policy_management` or `user_role_scope_management` capability.

```
GET /openapi/v1/app_scopes
```

Parameters:

Name	Type	Description
vrf_id	integer	Match app scopes by vrf_id.
root_app_scope_id	string	Match app scopes by root app scope id.
exact_name	string	Returns scope matching the exact name, case-sensitive.
exact_short_name	string	Returns scopes matching the exact short_name, case-sensitive.

Returns a list of scope objects.

Create a scope

This endpoint is used to create new scopes.

```
POST /openapi/v1/app_scopes
```

Parameters:

Name	Type	Description
short_name	string	User specified name of the scope.
description	string	User specified description of the scope.
short_query	JSON	Filter (or match criteria) associated with the scope.
parent_app_scope_id	string	ID of the parent scope.
policy_priority	integer	Default is 'last'. Used to sort workspace priorities. See Policy Ordering under Policies.

Sample python code

```
req_payload = {
   "short_name": "App Scope Name",
   "short_query": {
       "type":"eq",
       "field":"ip",
       "value": <....>
```

```
    },
    "parent_app_scope_id": <parent_app_scope_id>
}
resp = restclient.post('/app_scopes', json_body=json.dumps(req_payload))
```

To create a scope based on subnet, use the following short_query:

```
  "short_query":
    {
      "type":"subnet",
      "field":"ip",
      "value": "1.0.0.0/8"
    },
```

Get specific scope

This endpoint returns an instance of a scope.

`GET /openapi/v1/app_scopes/{app_scope_id}`

Returns the scope object associated with the specified ID.

Update a scope

This endpoint updates a scope. Changes to the `name` and `description` are applied immediately. Changes to the short_query mark the scope as 'dirty' and set the `dirty_short_query` attribute. Once all scope query changes, under a given root scope, are made, one needs to ping the Commit scope query changes endpoint to commit all the required updates.

`PUT /openapi/v1/app_scopes/{app_scope_id}`

Parameters:

Name	Type	Description
short_name	string	User specified name of the scope.
description	string	User specified description of the scope.
short_query	JSON	Filter (or match criteria) associated with the scope.

Returns the modified scope object associated with specified ID.

Delete a specific scope

This endpoint deletes the specified scope.

`DELETE /openapi/v1/app_scopes/{app_scope_id}`

If the Scope is associated with a workspace, Policy, User Inventory Filter, etc. this endpoint will return `422 Unprocessable Entity`. The returned Error object will contain a `details` attribute with the count of dependent objects along with the ids of the first 10 of each type. This information can be used to locate and remove the blocking dependencies.

Get scopes in policy priority order

This endpoint lists the scopes in the order that their corresponding primary workspace will be enforced.

```
GET /openapi/v1/app_scopes/{root_app_scope_id}/policy_order
```

Returns an array of scope objects.

Update the policy order

This endpoint will update the order at which policies are applied.

Warning This endpoint changes the order at which policies are applied. As a result new host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

```
POST /openapi/v1/app_scopes/{root_app_scope_id}/policy_order
```

Parameters:

Name	Type	Description
root_app_scope_id	string	Root scope or which the order is being changed.
ids	array	array of scope id strings in the order they should be enforced.

The `ids` array parameter must include all members of the root scope, including the root.

Commit scope query changes

This endpoint triggers an asynchronous background job to update all 'dirty' children under a given root scope. This job updates scopes and workspaces, see Scopes for more details.

```
POST /openapi/v1/app_scopes/commit_dirty
```

Parameters:

Name	Type	Description
root_app_scope_id	string	ID for a root scope for which all children will be updated.
sync	boolean	(optional) Indicate if the request should be synchronous.

Returns 202 to indicate the job has been enqueued. To check if the job has completed, poll the root scope's 'dirty' attribute to see if it has been set to false.

Users may pass the `sync` parameter to have the job run immediately. The request will return when done with a 200 status code. This request may take some time if many updates need to be applied.

Submit a group suggestion request

Submit a group suggestion request for a scope.

```
PUT /openapi/v1/app_scopes/{app_scope_id}/suggest_groups
```

Parameters: The request URL contains the following parameters

Name	Type	Description
app_scope_id	string	The unique identifier for the scope.

Parameters: The JSON query body contains the following keys

Name	Type	Description
start_time	string	Start time of the group suggestion input time interval.
end_time	string	End time of the group suggestion input time interval.

Response object: Returns an object with the following attributes:

Name	Type	Description
message	string	Message regarding success/failure in submission of group suggestion request.

Sample python code

```
app_scope_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'start_time': '2020-09-17T10:00:00-0700',
      'end_time': '2020-09-17T11:00:00-0700',
}
resp = restclient.put('/app_scopes/%s/suggest_groups' % app_scope_id,
                      json_body=json.dumps(req_payload))
```

Get group suggestion status

Query group suggestion status of the scope.

```
GET /openapi/v1/app_scopes/{app_scope_id}/suggest_groups_status
```

Parameters: The request URL contains the following parameters

Name	Type	Description
app_scope_id	string	The unique identifier for the scope.

Response object: Returns an object with the following attributes:

Name	Type	Description
status	string	Status of the group suggestion. Values: PENDING, COMPLETE, or FAILED

Sample python code

```
app_scope_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/app_scopes/%s/suggest_groups_status' % app_scope_id)
```

Configure Alerts

This set of APIs can be used to manage user alerts. They require the `user_alert_management` capability associated with the API key.

Alert Object

Each alert configuration object contains the following fields:

Attribute	Type	Description
app_name	string	Application name associated with the alert configuration.
rules	object	Set of conditions that must be met for the alert configuration to trigger an alert.
subjects	object	List of users who should receive the alert.
severity	string	Indicates the level of severity associated with an alert configuration.
individual_alert	boolean	Indicates whether individual alerts should be sent that triggers the alert configuration.
summary_alert_freq	string	Frequency of summary alerts to be sent for a particular alert configuration.
alert_type	string	Unique identifier of the alert configuration.

Attribute	Type	Description
app_instance_id	string	Unique identifier of a particular instance associated with the alert configuration

Get Alerts

This endpoint retrieves the list of alert configurations for a user. Alerts can be filtered to a given root scope. If no scope is provided, all alerts, for all scopes the user has access to, are returned. Service provider alerts will only be returned if the user is a site admin.

```
GET /openapi/v1/alert_confs
```

Parameters:

Name	Type	Description
root_app_scope_id	string	(optional) ID of a root scope to return alerts only assigned to that scope.

Response object: Returns a list of user alert objects.

Create an Alert

This endpoint is used to create a new alert.

```
POST openapi/v1/alert_confs
```

Parameters:

Attribute	Type	Description
app_name	string	Application name associated with the alert configuration.
rules	object	Set of conditions that must be met for the alert configuration to trigger an alert.
subjects	object	List of users who should receive the alert.
severity	string	Indicates the level of severity associated with an alert configuration.
individual_alert	boolean	Indicates whether individual alerts should be sent that triggers the alert configuration.

Attribute	Type	Description
summary_alert_freq	string	Frequency of summary alerts to be sent for a particular alert configuration.
alert_type	string	Unique identifier of the alert configuration.
app_instance_id	string	Unique identifier of a particular instance associated with the alert configuration.

The requesting user must have access to the provided scope. An alert without a scope is a 'Service Provider Alert' and only a site admin may create them.

Response object: Returns the newly created alert object.

Get Specific Alert

This endpoint returns a specific alert object.

```
GET /openapi/v1/alert_confs/
```

Parameters: The request URL contains the following parameters

Name	Type	Description
alert_id	string	Uniquely identifies the alert.

Response object: Returns an alert object associated with the specified ID.

Update an Alert

This endpoint is used to update an existing alert.

```
PUT /openapi/v1/alert_confs/
```

Parameters: The request URL contains the following parameters

Name	Type	Description
alert_id	string	To retrieve or modify the configuration settings for the alert.

The JSON request body contains the following parameters

Name	Type	Description
name	string	User specified name for the alert.
description	string	User specified description for the alert.

The requesting user must have access to the provided scope. A alert without a scope is called a 'Service Provider Alert' and only site admin may update them.

Response object: The updated alert object with the specified ID.

Delete Specific Alert

This endpoint deletes the specified alert.

```
DELETE /openapi/v1/alert_confs/{alert_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
alert_id	string	Uniquely identifies the alert.

Response object: None.

Roles

This set of APIs can be used to manage user roles. They require the `user_role_scope_management` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

Role object

The role object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the role.
app_scope_id	string	Scope to which the scope is defined, maybe empty for "Service Provider Roles".
name	string	User specified name for the role.
description	string	User specified description for the role.

Get roles

This endpoint returns a list of roles accessible to the user. Roles can be filtered to a given root scope. If no scope if provided, all roles, for all scopes the user has access to, are returned. Service provider roles will only be returned if the user is a site admin.

```
GET /openapi/v1/roles
```

Parameters:

Name	Type	Description
app_scope_id	string	(optional) ID of a root scope to return roles only assigned to that scope.

Response object: Returns a list of user role objects.

Sample python code

```
resp = restclient.get('/roles')
```

Create a role

This endpoint is used to create a new role.

```
POST /openapi/v1/roles
```

Parameters:

Name	Type	Description
name	string	User specified name for the role.
description	string	User specified description for the role.
app_scope_id	string	(optional) The scope ID under which the role is created If no scope ID mentioned the role is considered as service provider role.

The requesting user must have access to the provided scope. A role without a scope is called a 'Service Provider Role' and only site admin may create them.

Response object: Returns the newly created role object.

Sample python code

```
app_scope_id = '<app-scope-id>'
req_payload = {
      'name': 'Role Name',
      'description': 'Role Description',
      'app_scope_id': app_scope_id
}
restclient.post('/roles', json_body=json.dumps(req_payload))
```

Get specific role

This endpoint returns a specific role object.

```
GET /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
role_id	string	Uniquely identifies the role.

Response object: Returns a role object associated with the specified ID.

Sample python code

```
role_id = '<role-id>'
restclient.get('/roles/%s' % role_id)
```

Update a role

This endpoint is used to update an existing role.

```
PUT /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
role_id	string	Uniquely identifies the role.

The JSON request body contains the following parameters

Name	Type	Description
name	string	User specified name for the role.
description	string	User specified description for the role.

The requesting user must have access to the provided scope. A role without a scope is called a 'Service Provider Role' and only site admin may update them.

Response object: The updated role object with the specified ID.

Sample python code

```
role_id = '<role-id>'
req_payload = {
      'name': 'Role Name',
      'description': 'Role Description',
}
restclient.put('/roles/%s' % role_id, json_body=json.dumps(req_payload))
```

Give a role access to scope

This endpoint gives a role the specified access level to a scope.

```
POST /openapi/v1/roles/{role_id}/capabilities
```

Capabilities can only be added to the roles that the user has access to. If the roles is assigned to a scope, capabilities must correspond to that scope or its children. Service provider roles (those not assigned to a scope) can add capabilities for any scope.

Parameters: The request URL contains the following parameters

Name	Type	Description
role_id	string	Uniquely identifies the role.

The JSON request body contains the following parameters

Name	Type	Description
app_scope_id	string	ID of the scope to which access is provided.
ability	string	Possible values are SCOPE_READ, SCOPE_WRITE, EXECUTE, ENFORCE, SCOPE_OWNER, DEVELOPER

For more description of abilities, refer to Roles .

Response object:

Name	Type	Description
app_scope_id	string	ID of the scope to which access is provided.
role_id	string	ID of the role.
ability	string	Possible values are SCOPE_READ, SCOPE_WRITE, EXECUTE, ENFORCE, SCOPE_OWNER, DEVELOPER
inherited	boolean	

Sample python code

```
role_id = '<role-id>'
req_payload = {
  'app_scope_id': '<app-scope-id>',
  'ability': 'SCOPE_READ'
}
restclient.post('/roles/%s/capabilities' % role_id,
                json_body=json.dumps(req_payload))
```

Delete specific role

This endpoint deletes the specified role.

```
DELETE /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
role_id	string	Uniquely identifies the role.

Response object: None.

Sample python code

```
role_id = '<role-id>'
restclient.delete('/roles/%s' % role_id)
```

Users

This set of APIs manages users. They require the `user_role_scope_management` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

User object

The user object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the user role.
email	string	Email associated with user account.
first_name	string	First name.
last_name	string	Last name.
app_scope_id	string	The scope to which the user is assigned. Maybe empty if the user is a “Service Provider User”.
role_ids	list	List of IDs of roles assigned to the user account.
by-pass_external_auth	boolean	True for local users and false for external auth users (ldap or sso).
disabled_at	integer	Unix timestamp of when the user has been disabled. Zero or null, otherwise.

Get users

This endpoint returns a list of user objects known to the Secure Workload appliance.

```
GET /openapi/v1/users
```

Parameters: The request URL contains the following parameters

Name	Type	Description
include_disabled	boolean	(optional) To include disabled users, defaults to false.
app_scope_id	string	(optional) Return only users assigned to the provided scope.

Response object: Returns a list of user objects. Only site admins can see 'Service provider users', i.e. those not assigned to a scope.

Sample python code

```
    GET /openapi/v1/users
```

Create a new user account

This endpoint is used to create a new user account.

```
POST /openapi/v1/users
```

Parameters: The JSON request body contains the following parameters

Name	Type	Description
email	string	Email associated with user account.
first_name	string	First name.
last_name	string	Last name.
app_scope_id	string	(optional) Root scope to which user belongs.
role_ids	list	(optional) The list of roles that should be assigned to the user.

The app_scope_id is the ID of the root scope to which the user is to be assigned. If the app_scope_id is not present then the user is a 'Service Provider user.' Only site admins can create service provider users. The `role_ids` are the ids of the roles that were created under the specified app scope.

Response object: Returns the newly created user object.

Sample python code

```
req_payload = {
    "first_name": "fname",
    "last_name": "lname",
```

```
    "email": "foo@bar.com"
    "app_scope_id": "root_appscope_id",
    "role_ids": ["roleid1", "roleid2"]
}
resp = restclient.post('/users', json_body=json.dumps(req_payload))
```

Get specific user

This endpoint returns specific user object.

```
GET /openapi/v1/users/{user_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object.

Response object: Returns a user object associated with specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/users/%s' % user_id)
```

Update a user

This endpoint updates an existing user.

```
PUT /openapi/v1/users/{user_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being updated.

The JSON request body contains the following parameters

Name	Type	Description
email	string	Email associated with user account.
first_name	string	First name.
last_name	string	Last name.
app_scope_id	string	Root App Scope ID (only allowed for site admins)

Response object: Returns the newly updated user object.

Sample python code

```
req_payload = {
    "first_name": "fname",
    "last_name": "lname",
    "email": "foo@bar.com"
```

```
    "app_scope_id": "root_appscope_id",
}
restclient.put('/users', json_body=json.dumps(req_payload))
```

Enable/reactivate a deactivated user

This endpoint is used to re-enable a deactivated user.

```
POST /openapi/v1/users/{user_id}/enable
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being enabled.

Response object: Returns the reactivated user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.post('/users/%s/enable' % user_id)
```

Add role to the user account

This endpoint is used to add a role to a user account.

```
PUT /openapi/v1/users/{user_id}/add_role
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being modified.

The JSON request body contains the following parameters

Name	Type	Description
role_id	string	ID of the role object to be added.

Response object: Returns the modified user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
    "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.put('/users/%s/add_role' % user_id,
                      json_body=json.dumps(req_payload))
```

Remove role from the user account

This endpoint is used to remove a role from a user account.

```
DELETE /openapi/v1/users/{user_id}/remove_role
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being deleted.

The JSON request body contains the following parameters

Name	Type	Description
role_id	string	ID of the role object to be removed.

Response object: Returns the modified user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
    "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.delete('/users/%s/remove_role' % user_id,
                         json_body=json.dumps(req_payload))
```

Delete specific user

This endpoint deletes the specified user account.

```
DELETE /openapi/v1/users/{user_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being deleted.

Response object: Returns the deleted user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.delete('/users/%s' % user_id)
```

Inventory filters

Inventory filters encode the match criteria for inventory search queries. This set of APIs provide functionality similar to what is described in Inventory Filters . They require either `sensor_management or app_policy_management` capability associated with the API key.

Inventory Filter Object

The inventory filter JSON object is returned as a single object or an array of objects depending on the API endpoint. The object's attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the inventory filter.
name	string	User specified name of the inventory filter.
app_scope_id	string	ID of the scope associated with the filter.
short_query	JSON	Filter (or match criteria) associated with the filter.
primary	boolean	When 'true' the filter is restricted to the ownership scope.
public	boolean	When 'true' the filter provides a service for its scope. Must also be primary or scope restricted.
query	JSON	Filter (or match criteria) associated with the filter in conjunction with the filters of the parent scopes. These conjunctions take effect if 'restricted to ownership scope' checkbox is checked. If 'primary' field is false then query is same as short_query.

Get inventory filters

This endpoint returns a list of inventory filters visible to the user.

```
GET /openapi/v1/filters/inventories
```

Parameters:

Name	Type	Description
vrf_id	integer	Match inventory filters by vrf id.
root_app_scope_id	string	Match inventory filters by root app scope id.
name	string	Returns inventory filters matching part of the name, case-insensitive.

Name	Type	Description
exact_name	string	Returns inventory filters matching the exact name, case-sensitive.

Create an inventory filter

This endpoint is used to create an inventory filter.

```
POST /openapi/v1/filters/inventories
```

Parameters:

Name	Type	Description
name	string	User specified name of the application scope.
query	JSON	Filter (or match criteria) associated with the filter.
app_scope_id	string	ID of the scope associated with the filter.
primary	boolean	When ‘true’ the filter is restricted to the ownership scope.
public	boolean	When ‘true’ the filter provides a service for its scope. Must also be primary/scope restricted.

Sample python code

```
req_payload = {
  "app_scope_id": <app_scope_id>,
  "name": "sensor_config_inventory_filter",
  "query": {
    "type": "eq",
    "field": "ip",
    "value": <sensor_interface_ip>
  },
}
resp = restclient.post('/filters/inventories', json_body=json.dumps(req_payload))
```

Validate an inventory filter query

This endpoint will validate a query’s structure against the required schema.

```
POST /openapi/v1/filters/inventories/validate_query
```

Parameters:

Name	Type	Description
query	JSON	Filter (or match criteria) associated with the scope.

Response object:

Attribute	Type	Description
valid	boolean	Indicates if the query is valid
errors	array	If invalid, details about the errors

Get specific inventory filter

This endpoint returns an instance of an inventory filter.

```
GET /openapi/v1/filters/inventories/{inventory_filter_id}
```

Returns an inventory filter object associated with specified ID.

Update specific inventory filter

This endpoint is used to update an inventory filter.

```
PUT /openapi/v1/filters/inventories/{inventory_filter_id}
```

Parameters:

Name	Type	Description
name	string	User specified name of the scope.
query	JSON	Filter (or match criteria) associated with the scope.
app_scope_id	string	ID of the scope associated with the filter.
primary	boolean	When ‘true’ the filter is restricted to the ownership scope.
public	boolean	When ‘true’ the filter provides a service. May be used as part of policy generation. Must also be primary/scope restricted.
Usages	boolean	Collection of member policy and configuration statistics.

Delete a specific inventory filter

This endpoint deletes the specified inventory filter.

```
DELETE /openapi/v1/filters/inventories/{inventory_filter_id}
```

Flow Search

The flow search feature provides similar functionality as described in Flows. These set of APIs require the `flow_inventory_query` capability associated with the API key.

Query for Flow Dimensions

This endpoint returns the list of flow columns on which search criteria (or *filters*) can be specified for flow search queries (below). For more information on column descriptions, see Columns and Filters.

```
GET /openapi/v1/flowsearch/dimensions
```

Parameters: None

Response object:

Name	Type	Description
dimensions	List of strings	List of user uploaded and orchestrator dimensions.

Sample python code

```
restclient.get('/flowsearch/dimensions')
```

Query for Flow Metrics

This endpoint returns the list of metrics, for example, byte count and packet count, associated with flow observations.

```
GET /openapi/v1/flowsearch/metrics
```

Parameters: None

Response object:

Name	Type	Description
metrics	List of strings.	List of available metrics.

Sample python code

```
restclient.get('/flowsearch/metrics')
```

Query for Flows

This endpoint returns the list of flows matching the filter criteria. Each flow object in the result has attributes that are a union of flow dimensions (returned by the flow dimensions API above) as well as the flow metrics (returned by the flow metrics API above).

`POST /openapi/v1/flowsearch`

The list of columns that can be specified in the filter criteria can be obtained by `/openapi/v1/flowsearch/dimensions` API.

Parameters: The query body consists of a JSON body with the following keys.

Name	Type	Description
t0	integer or string	Flow search start time (epoch or ISO 8601)
t1	integer or string	Flow search end time (epoch or ISO 8601)
filter	JSON	Query filter. If filter is empty (i.e. {}), query matches all flows.
scopeName	string	Full name of the scope to which query is restricted.
dimensions	array	(Optional) List of dimension names to be returned in the result of flowsearch API. This is an optional parameter. If unspecified, flowsearch results return all the available dimensions. This option is useful to specify a subset of the available dimensions when caller does not care about the rest of the dimensions.
metrics	array	(Optional) List of metric names to be returned in the result of flowsearch API. This is an optional parameter. If unspecified, flowsearch results return all the available metrics. This option is useful to specify a subset of the available metrics when caller does not care about the rest of the metrics.
limit	integer	(Optional) Number of response flows limit.
offset	string	(Optional) Offset object received from previous response.

Name	Type	Description
descending	boolean	(Optional) If this parameter is false or left unspecified, results are in ascending order of timestamps. If parameter value is true, results are in descending order of timestamps.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "t0": "2016-06-17T09:00:00-0700",
    "t1": "2016-06-17T17:00:00-0700",
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "dst_hostname",
                "value": "prod"
            },
            {
                "type": "in",
                "field": "dst_port",
                "values": ["80", "443"]
            }
        ]
    },
    "scopeName": "Default:Production:Web",
    "limit": 100,
    "offset": <offset-object>
}
```

Filters

The filter supports primitive filters and logical filters (“not”, “and”, “or”) comprised of one or more primitive filters. Format of primitive filter is as follows:

```
{"type" : "<OPERATOR>", "field": "<COLUMN_NAME>", "value": "<COLUMN_VALUE>"}
```

For primitive filters, operator can be a comparison operator like `eq, ne, lt, lte, gt` or `gte`. Operator could also be `in, regex, subnet, contains` or `range`.

Some examples of primitive filters might include:

```
{"type": "eq", "field": "src_address", "value": "7.7.7.7"}

{"type": "regex", "field": "src_hostname", "value": "prod.*"}

{"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}

# Note, 'in' clause uses 'values' key instead of 'value'
{"type": "in", "field": "src_port", "values": [80, 443]}
```

You can also specify complex filters using boolean operations like not, and or or. Following are some examples of these type of filters:

```
# "and" and "or" operators need to specify list of "filters"
{"type": "and",
    "filters": [
        {"type": "in", "field": "src_port", "values": [80, 443]},
        {"type": "regex", "field": "src_hostname", "value": "prod.*"}
    ]
}

# "not" operator needs to specify a "filter"
{"type": "not",
    "filter": {"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}
}
```

More formally, schema of `filter` in the flow search request is as follows:

Keys	Values
type	Filter type
field	Filter field column for primitive filters
filter	Filter object (only used for `not` filter type)
filters	List of filter objects (used for `and` and `or` filter types)
value	Value for primitive filters
values	List of values for primitive filters with filter type `in` or `range`

Primitive Filter Types

eq, ne—Searches flows for equality or inequality respectively in column specified by "`field`" with value specified by "`value`". Supports the following fields: `src_hostname, dst_hostname, src_address, dst_address, src_port, dst_port, src_scope_name, dst_scope_name, vrf_name, src_enforcement_epg_name, dst_enforcement_epg_name, proto`. These operators also work on user labelled columns.

lt, lte, gt, gte—Searches flows where values of column specified by "`field`" are less than, less than equal to, greater than or greater than equal to (as applicable) the value specified by "`value`". Supports the following fields: `[src_port, dst_port]`.

range—Searches flows for values of column specified by "field" between range start and range end specified by "values" list (this list must be of size 2 for *"range"* filter type – first value is the range start and second is the range end). Supports the following fields: [src_port, dst_port].

in—Searches flows for membership in column specified by "`field`" with membership list specified by "`values`". Supports the following fields: `src_hostname, dst_hostname, src_address, dst_address, src_port, dst_port, src_scope_name, dst_scope_name, vrf_name, src_enforcement_epg_name, dst_enforcement_epg_name, proto`. This operator also works on user labelled columns.

regex, contains—Searches flows for regex matches or containment matches respectively in column specified by "`field`" with regex specified by "`value`". Supports the following fields: `src_hostname, dst_hostname, src_scope_name, dst_scope_name, vrf_name, src_enforcement_epg_name, dst_enforcement_epg_name`. These operators also work on user labelled columns. Filters with `regex` type must use Java style regex patterns as "`value`".

subnet—Searches flows for subnet membership specified by "`field`" as a string in CIDR notation. Supports the following fields: `["src_address", "dst_address"]`

Logical Filter Types

- **not**—Logical "not" filter of object specified by "`filter`".
- **and**—Logical "and" filter of list of filter objects specified by "`filters`".
- **or**—Logical "or" filter of list of filter objects specified by "`filters`".

Response object:

Keys	Values
offset	Response offset to be passed for the next page of results
results	List of results

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

Sample python code

```
req_payload = {"t0": "2016-11-07T09:00:00-0700",
               "t1": "2016-11-07T19:00:00-0700",
               "scopeName": "Default:Prod:Web",
               "limit": 10,
               "filter": {"type": "and",
                   "filters": [
                     {"type": "subnet", "field": "src_address", "value": "1.1.11.0/24"},

                        {"type": "regex", "field": "src_hostname", "value": "web*"}
                     ]
               }
 }

 resp = restclient.post('/flowsearch', json_body=json.dumps(req_payload))
 print resp.status_code
 if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

TopN Query for Flows

This endpoint returns a top N sorted list of values of specified dimension where rank in the list is determined by the aggregate of specified metric.

`POST /openapi/v1/flowsearch/topn`

Parameters:

The list of columns that can be specified in the filter criteria can be obtained by `/openapi/v1/flowsearch/dimensions` API. The body of the request should be a JSON formatted query. An example of a query body is shown below. Parameters `t0` and `t1` in the request body can be in epoch format or in ISO 8601 format. TopN API only allows querying maximum time range of one day. The dimension on which the grouping has to be done should be specified through `dimension`. The metric by which top N results

need to ranked should be specified in `metric` field in the JSON body. You should specify a `threshold` with a minimum value of 1 which signifies the 'N' in 'TopN'. The maximum value of this `threshold` is 1000. Even if the user specify more than 1000 the API returns only a maximum of 1000 results. In addition, you must specify a parameter called `scopeName` which is the full name of the scope to which you want to restrict the search. The `filter` is same as that of filter of Flow Search Filters, on page 80 . If the `filter` is not mentioned, the topN is applied on all the flow entries.

```
{
  "t0": "2016-06-17T09:00:00-0700",    # t0 can also be 1466179200
  "t1": "2016-06-17T17:00:00-0700",    # t1 can also be 1466208000
  "dimension": "src_address",
  "metric": "fwd_pkts",
 "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},  #optional

  "threshold": 5,
  "scopeName": "Default"
}
```

The query body consists of a JSON body with the following keys.

Keys	Values
t0	Start time of the Flow (epoch or ISO 8601)
t1	End time of the Flow (epoch or ISO 8601)
filter	Query filter. If filter is empty (i.e. {}),or filter is absent (optional) then topN query is applied on all flow entries
scopeName	Full name of the scope to which query is restricted to
dimension	The dimension is a field on which we are grouping.
metric	The metric is the total count of values of the dimension.
threshold	Threshold is N in the topN.

Response object:

Keys	Values
result	Array of the top N entries

Sample python code

```
req_payload = {
    "t0": "2017-06-07T08:20:00-07:00",
    "t1": "2017-06-07T14:20:00-07:00",
    "dimension": "src_address",
    "metric": "fwd_pkts",
    "filter": {"type": "ne", "field": "src_address", "value": "172.29.203.193"},
    "threshold": 5,
    "scopeName": "Default"
}
```

```
resp = rc.post('/flowsearch/topn',
               json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
[
  { "result": [
    {"src_address": "172.31.239.163", "fwd_pkts": 23104},
    {"src_address": "172.31.239.162", "fwd_pkts": 22410},
    {"src_address": "172.31.239.166", "fwd_pkts": 16185},
    {"src_address": "172.31.239.168", "fwd_pkts": 15197},
    {"src_address": "172.31.239.169", "fwd_pkts": 15116}
    ]
  }
]
```

Flow Count

This endpoint returns the number of flow observations matching the specified criteria.

`POST /openapi/v1/flowsearch/count`

Parameters:

The body of the request should be a JSON formatted query. An example of a query body is shown below. Parameters `t0` and `t1` in the request body can be in epoch format or in ISO 8601 format. This API only allows querying maximum time range of one day. In addition, you need to specify the `scopeName` parameter which is the full name of the scope to which you want to restrict the search. If this parameter is not specified, flow observation count API request applies to all scopes to which you have read access. The `filter` is same as that of filter of Flow Search Filters .

```
{
  "t0": "2016-06-17T09:00:00-0700",    # t0 can also be 1466179200
  "t1": "2016-06-17T17:00:00-0700",    # t1 can also be 1466208000
  "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},
  "scopeName": "Default"
}
```

The query body consists of a JSON body with the following keys.

Keys	Values
t0	Start time of the flow (epoch or ISO 8601)
t1	End time of the flow (epoch or ISO 8601)
filter	Query filter. If filter is empty (i.e. {}) then query matches all flows.
scopeName	Full name of the scope to which query is restricted to

Response object:

Keys	Values
count	The number of flow observations matching the flow search criteria.

Sample python code

```
req_payload = {
      "t0": "2017-07-20T08:20:00-07:00",
      "t1": "2017-07-20T10:20:00-07:00",
      "scopeName": "Tetration",
      "filter": {
                  "type": "eq",
                  "field": "dst_port",
                  "value": "5642"
          }
}
resp = rc.post('/flowsearch/count',
               json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
{"count":508767}
```

Inventory

The inventory search APIs provide similar functionality as described in inventory search. These set of APIs require the `flow_inventory_query` capability associated with the API key.

Query for inventory dimensions

This endpoint returns the list of inventory columns on which search criteria (or *filters*) can be specified for inventory search queries.

`GET /openapi/v1/inventory/search/dimensions`

Inventory search

This endpoint returns the list of inventory items matching the specified criteria.

`POST /openapi/v1/inventory/search`

The list of columns that can be specified in the filter criteria can be obtained with the /openapi/v1/inventory/search/dimensions API.

Parameters:

Name	Type	Description
filter	JSON	A filter query.

Name	Type	Description
scopeName	string	(optional) Name of the scope by which to limit results.
limit	integer	(optional) Max number of results to return.
offset	integer	(optional) Offset from the previous request to get the next page.

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
    "filter": {
        "type": "contains",
        "field": "hostname",
        "value": "collector"
    },
    "scopeName": "Default:Production:Web",  // optional
    "limit": 100,
    "offset": "<offset-object>"               // optional
}
```

To get the different types of filters supported refer to Filters, on page 80

The query body consists of a JSON body with the following keys.

Keys	Values
filter	Query filter. If filter is empty (i.e. {}), then query matches all inventory items.
scopeName	Full name of the scope to which query is restricted to (optional)
dimensions	List of dimension names to be returned in the result of inventory search API. This is an optional parameter. If unspecified, results return all the available dimensions. This option is useful to specify a subset of the available dimensions when caller does not care about the rest of the dimensions.
limit	Number of response items limit (optional)
offset	Offset object received from previous response (optional)

Response

The response is a JSON object in the body with the following properties.

Name	Type	Description
offset	integer	Response offset to be passed for the next page of results.

Name	Type	Description
results	array of objects	List of results.

The response may contain an `offset` field for paginated responses. Users will need to specify the same offset in the subsequent request to get the next set of results.

Sample Python code

```
req_payload = {
  "scopeName": "Tetration", # optional
  "limit": 2,
  "filter": {"type": "and",
     "filters": [
        {"type": "eq", "field": "vrf_name", "value": "Tetration"},
        {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
        {"type": "contains", "field": "hostname", "value": "collector"}
     ]
  }
}

resp = restclient.post('/inventory/search', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Inventory Statistics

This endpoint returns statistics for inventory items.

```
GET /openapi/v1/inventory/{id}/stats?t0=<t0>&t1=<t1>&td=<td>
```

Table 6:

Path Parameter	Description
id	Inventory item id as `{ip}-{vrf_id}` such as `1.1.1.1-123`

Query Parameter	Description
t0	Start time for statistics in epoch time
t1	End time for statistics in epoch time
td	Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as “minute”, “hour”, and “day”.

Sample Python code

```
resp = restclient.get('/inventory/1.1.1.1-123/stats?t0=1483228800&t1=1485907200&td=day')
```

Inventory count

This endpoint returns the count of inventory items matching the specified criteria.

`POST /openapi/v1/inventory/count`

The list of columns that can be specified in the filter criteria can be obtained with the `/openapi/v1/inventory/search/dimensions` API.

Parameters:

Name	Type	Description
filter	JSON	A filter query.
scopeName	string	(optional) Name of the scope by which to limit results.

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "hostname",
                "value": "prod"
            },
            {
                "type": "subnet",
                "field": "ip"
                "value": "6.6.6.0/24"
            }
        ]
    },
    "scopeName": "Default:Production:Web",  # optional
}
```

Response

The response is a JSON object in the body with the following properties.

Table 7:

Keys	Values
count	Number of inventory items matching the filter Criteria

Sample python code

```
req_payload = {
  "scopeName": "Tetration", # optional
  "filter": {"type": "and",
     "filters": [
        {"type": "eq", "field": "vrf_name", "value": "Tetration"},
        {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
        {"type": "contains", "field": "hostname", "value": "collector"}
     ]
```

```
    }
  }

  resp = restclient.post('/inventory/count', json_body=json.dumps(req_payload))
  print resp.status_code
  if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Inventory vulnerability

This endpoint returns CVEs corresponding to IP addresses associate with vulnerable workloads.

This API is only available to users with a minimum read access to root scope.

```
POST /openapi/v1/inventory/cves/{rootScopeID}
```

Parameters:

Name	Type	Description
ips	list of strings	List of IPs to fetch CVE information.

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
    {
        "ips": [
            "10.18.187.72",
            "10.18.187.73"
        ]
    }
```

Response

The response is an array of JSON objects in the body with the following properties.

Name	Type	Description
ip	string	IP address
cve_ids	list of strings	List of CVE IDs on the inventory with the ip address.

Sample Python code

```
  root_scope_id = "5fa0d242497d4f7d968c669b"
  req_payload = {
    "ips":["10.18.187.72", "10.18.187.73"]
  }

  resp = restclient.post('/inventory/cves/' + root_scope_id,
json_body=json.dumps(req_payload))
  print resp.status_code
  if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Workload

The workload APIs provides programmatic access to the contents of the Workload Profile page. This set of APIs requires `sensor_management or flow_inventory_query` capability associated with the API key.

Workload details

This endpoint returns the specific workload given agent UUID.

```
GET /openapi/v1/workload/{uuid}
```

Path Parameter	Description
uuid	Agent UUID

Response

The response is a workload object associated with the specified UUID. The workload object's attributes schema is described below:

Table 8:

Attribute	Type	Description
agent_type	integer	Agent type in enum
agent_type_str	string	Agent type in plain text
auto_upgrade_opt_out	boolean	If true, agents do not get automatically upgraded on cluster upgrade
cpu_quota_mode	integer	CPU quota control
cpu_quota_us	integer	CPU quota usage
current_sw_version	string	Version of agent software running on the workload
data_plane_disabled	boolean	If true, flow telemetry data is not exported from the agent to the cluster
desired_sw_version	string	Version of agent software intended to be running on the workload
enable_cache_sidechannel	boolean	If true, side channel attack detection is enabled
enable_forensics	boolean	If true, forensics is enabled
enable_meltdown	boolean	If true, meltdown exploit detection is enabled

Attribute	Type	Description
enable_pid_lookup	boolean	If true, process lookup is enabled
forensics_cpu_quota_mode	integer	Forensics CPU quota control
forensics_cpu_quota_us	integer	Forensics quota usage
forensics_mem_quota_bytes	integer	Forensics memory quota in bytes
host_name	string	Host name on the workload
interfaces	array	Array of Interface objects
kernel_version	string	Kernel version
last_config_fetch_at	integer	Last config fetched at
last_software_update_at	integer	Last software is the timestamp at which agent reported its current version
max_rss_limit	integer	Max memory limit
platform	string	Platform of the workload
uuid	string	Unique ID of the agent
windows_enforcement_mode	string	Type of Windows enforcement mode, WAF(Windows Advanced Firewall) or WFP(Windows Filtering Platform)

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s' % (agent_uuid))
```

Workload Statistics

This endpoint returns statistics for a workload.

```
GET /openapi/v1/workload/{uuid}/stats?t0=<t0>&t1=<t1>&td=<td>
```

Path Parameter	Description
uuid	Agent UUID

The query URL contains the following parameters

Query Parameter	Description
t0	Start time for statistics in epoch time

Query Parameter	Description
t1	End time for statistics in epoch time. The end time cannot exceed the start time by more . than a day.
td	Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as “minute”, “hour”, and “day”.

Response

The response is a JSON object in the body with the following properties.

Name	Type	Description
timestamp	string	Time at which metrics were gathered (epoch or ISO 8601)
results	object	Metrics

Metrics is a JSON object with the following properties

Name	Type	Description
flow_count	integer	Number of flows.
rx_byte_count	integer	Number of received bytes.
rx_packet_count	integer	Number of received packets.
tx_byte_count	integer	Number of transmitted bytes.
tx_packet_count	integer	Number of transmitted packets.

Sample Python code

```
 agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
 td = 15 * 60 # 15 minutes
 resp = restclient.get('/workload/%s/stats?t0=1483228800&t1=1485907200&td=%d' % (agent_uuid,
td))

 # This code queries workload statistics for a week
 t0 = 1483228800
 for _ in range(7):
   t1 = t0 + 24 * 60 * 60
   resp = restclient.get('/workload/%s/stats?t0=%d&t1=%d&td=day' % (agent_uuid, t0, t1))
   t0 = t1
```

Installed Software Packages

This endpoint returns list of packages installed on the workload.

```
GET /openapi/v1/workload/{uuid}/packages
```

Path Parameter	Description
uuid	Agent UUID

Response

The respone is an array of package JSON objects. The package object's schema is described below:

Attribute	Type	Description
architecture	string	Architecture of the package
name	string	Name of the package
publisher	string	Publisher of the package
version	string	Version of the package

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/packages' % (agent_uuid))
```

Workload Vulnerabilities

This endpoint returns list of vulnerabilities observed on the workload.

```
GET /openapi/v1/workload/{uuid}/vulnerabilities
```

The vulnerabilities object consists of a JSON body with the following keys.

Path Parameter	Description
uuid	Agent UUID

Response

The response is an array of vulnerability JSON objects. The vulnerability object's schema is described below:

Attribute	Type	Description
cve_id	string	Common Vulnerability Exposure ID
package_infos	array	Array of Package Info objects
v2_score	float	CVSS V2 Score
v2_access_complexity	string	CVSS V2 Access Compleixty
v2_access_vector	string	CVSS V2 Access Vector
v2_authentication	string	CVSS V2 Authentication
v2_availability_impact	string	CVSS V2 Availability Impact

Attribute	Type	Description
v2_confidentiality_impact	string	CVSS V2 Confidentiality Impact
v2_integrity_impact	string	CVSS V2 Intergrity Impact
v2_severity	string	CVSS V2 Severity
v3_score	float	CVSS V3 Score
v3_attack_complexity	string	CVSS V3 Attack Compleixty
v3_attack_vector	string	CVSS V3 Attack Vector
v3_availability_impact	string	CVSS V3 Availability Impact
v3_base_severity	string	CVSS V3 Base Severity
v3_confidentiality_impact	string	CVSS V2 Confidentiality Impact
v3_integrity_impact	string	CVSS V3 Intergrity Impact
v3_privileges_required	string	CVSS V3 Privileges Required
v3_scope	string	CVSS V3 Scope
v3_user_interaction	string	CVSS V3 User Interaction
cvm_score	float	Cisco Security Risk Score
cvm_severity	string	Cisco Security Risk Score Severity
cvm_easily_exploitable	bool	Cisco Security Risk Score Easily Exploitable
cvm_malware_exploitable	bool	Cisco Security Risk Score Malware Exploitable
cvm_active_internet_breach	bool	Cisco Security Risk Score Active Internet Breach
cvm_popular_target	bool	Cisco Security Risk Score Popular Target
cvm_predicted_exploitable	bool	Cisco Security Risk Score Predicted Exploitable
cvm_fix_available	bool	Cisco Security Risk Score Fix Available

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/vulnerabilities' % (agent_uuid))
```

Workload Long Running Processes

This endpoint returns list of long running processes on the workload. Long running processes are defined as processes that have at least 5 minutes uptime.

```
GET /openapi/v1/workload/{uuid}/process/list
```

Path Parameter	Description
uuid	Agent UUID

Response

The response is a list of processes JSON objects.

Attribute	Type	Description
cmd	string	Command string of the process
binary_hash	string	Sha256 of the process binary in hex
ctime	long	ctime of the process binary in us
mtime	long	mtime of the process binary in us
exec_path	string	Process executable path
exit_usec	long	Time when the process exited in us
num_libs	integer	Number of libs the process loads
pid	integer	Process ID
ppid	integer	Parent process ID
pkg_info_name	string	Name of the package associated with the process
pkg_info_version	string	Version of the package associated with the process
proc_state	string	Process state
uptime	long	Uptime of the process in us
username	string	Username of the process
resource_usage	array	Array of Resource Usage object

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/openapi/v1/workload/%s/process/list' % (agent_uuid))
```

Workload Process Snapshot Summary

This endpoint returns process snapshot summary on this workload. A process snapshot contains all the processes that are captured by the workload at a given time. Currently one copy of the latest process snapshot is retained. The endpoint supports POST method with empty payload to enable easier future expansion.

```
POST /openapi/v1/workload/{uuid}/process/tree/ids
```

Path Parameter	Description
uuid	Agent UUID

Response

The response is a list of process snapshot summary JSON objects.

Attribute	Type	Description
sensor_uuid	string	Agent UUID
handle	string	Handle to the process snapshot to be retrieved
process_count	integer	Number of processes in the snapshot
ts_usec	integer	Timestamp when the snapshot is captured

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
                       agent_uuid, json_body=json.dumps(payload))
```

Workload Process Snapshot

This endpoint returns process snapshot on this workload. A process snapshot contains all the processes that are captured by the workload at a given time. Currently one copy of the latest process snapshot is retained. This endpoint needs to be used together with the workload process snapshot summary endpoint.

```
POST /openapi/v1/workload/{uuid}/process/tree/details
```

Path Parameter	Description
uuid	Agent UUID

Payload Field	Type	Description
handle	string	Handle to the process snapshot to be retrieved

Response

The response is a list of processes belonging to the snapshot in JSON.

Attribute	Type	Description
command_string	string	Tokenized command string
command_string_raw	string	Raw command string
binary_hash	string	Sha256 of the process binary in hex
ctime	long	ctime of the process binary in us
mtime	long	mtime of the process binary in us
exec_path	string	Process executable path
process_id	integer	Process ID
parent_process_id	integer	Parent process ID
process_key	integer	Unique key to the process
parent_process_key	integer	Unique key to the parent process
pkg_info_name	string	Name of the package associated with the process
pkg_info_version	string	Version of the package associated with the process
proc_state	string	Process state
uptime	long	Uptime of the process in us
username	string	Username of the process
cve_ids	array	Array of CVEID object

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
            agent_uuid, json_body=json.dumps(payload))
handle = json.loads(resp.text)['process_summary'][0]['summary'][0]['handle']
payload = {
  "handle": handle,
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/details' %
            agent_uuid, json_body=json.dumps(payload))
```

JSON Object Definitions

Interface

Attribute	Type	Description
ip	string	IP Address of the interface
mac	string	Mac Address of the interface
name	string	Name of the interface
netmask	string	Netmask of the interface
pcap_opened	boolean	If false, packet captures are not enabled for the interface
tags_scope_id	array	Scope IDs associated with the interface
vrf	string	VRF Name
vrf_id	integer	VRF ID

Package Info

Attribute	Type	Description
name	string	Package name
version	string	Package version

Resource Usage

Attribute	Type	Description
cpu_usage	float	CPU usage
memory_usage_kb	integer	Memory usage
ts_usec	long	Timestamp in us when the resource usage is captured

CVE ID

Attribute	Type	Description
cve_id	string	cve ID
impact__cvss_v2__access_complexity	string	CVE access complexity
impact__cvss_v2__access_vector	string	CVE access vector

Default Policy Generation Config

This set of APIs is used to read and update the default policy generation config for a root scope.

The APIs require the `app_policy_management` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

- Policy Generation Config object, on page 99
- Get the Default Policy Generation Config, on page 100
- Set the Default Policy Generation Config, on page 100

Policy Generation Config object

Attribute	Type	Description
carry_over_policies	boolean	Any policy marked as approved will be maintained, if possible
deep_policy_generation	boolean	creates policies for the whole scope tree under give scope, includes all the members in the give scope
skip_clustering	boolean	set to true to skip clustering, will generate policies with existing approved clusters and scopes
auto_accept_policy_connectors	boolean	auto accepts all outgoing policy connectors
enable_exclusion_filter	boolean	apply exclusion filters to input flow data
enable_default_exclusion_filter	boolean	apply default exclusion filters to input flow data
remove_redundant_policies	boolean	remove redundant policies during deep policy generation
enable_service_discovery	boolean	setting false skips policy generation based on ephemeral port range in adm pipeline reported by the sensor, currently used for generating policies for Windows Active Directory.
externals	array	ordered list of external dependency objects
clustering_granularity	string	one of VERY_COARSE, COARSE, MEDIUM, FINE, VERY_FINE

Attribute	Type	Description
policy_compression	string	one of DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, VERY_AGGRESSIVE
port_generalization	string	one of DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, VERY_AGGRESSIVE
sim_policy	integer	1 => flows, 2 => processes, 5 => both

The External Dependency object

Name	Type	Description
id	string	id of the filter
filter_type	string	AppScope or UserInventoryFilter
incluce	array	object with user_filters boolean to enable and user_filter_list for the ordered list of provided service inventory filters

Get the Default Policy Generation Config

This endpoint returns the current default policy generation config. Can return an empty object if none has been created.

```
GET /openapi/v1/app_scopes/default_adm_run_config
```

Parameters:

The request URL contains the following parameters

Name	Type	Description
root_app_scope_id	string	The unique identifier of the root scope to which this default config applies

Response object: Returns the current default policy generation config or an empty object if none has been created

Set the Default Policy Generation Config

This endpoint sets the default policy generation config.

```
PUT /openapi/v1/app_scopes/default_adm_run_config
```

Parameters in addition to the values of the policy generation config object list above.

Name	Type	Description
root_app_scope_id	string	The unique identifier of the root scope to which this default config applies

Response object: Returns the default policy generation config.Response object: Returns the default policy generation config.

Forensics Intent

The software agents APIs are associated with managing forensic intents.

Forensic intents link a forensic profile with the group of agents it applies to. The group of agents is defined using an inventory filter.

These set of APIs require the `sensor_management` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

Forensic intent object

Attribute	Type	Description
id	string	unique identifier of the intent
name	string	name of the intent
inventory_filter_id	array	id of the inventory filter associated with the intent
forensic_config_profile_id	integer	id of the profile associated with this intent
created_at	integer	Unix timestamp of when the intent was created
updated_at	integer	Unix timestamp of when the intent was last updated

Listing a forensic intents

This endpoint lists all existing forensic profiles

```
GET /openapi/v1/inventory_config/forensic_intents
```

Parameters: None

This endpoint returns an array of forensic intent object summaries.

Retrieving a Single Forensic Intent

```
GET /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Parameters:

Name	Type	Description
intent_id	string	id of the intent

Returns a detailed representation of the forensic intent object.

Creating a Forensic Intent

```
POST /openapi/v1/inventory_config/forensic_intents
```

Parameters:

Name	Type	Description
name	string	name of the intent
inventory_filter_id	string	id of the inventory filter associated with the intent
forensic_config_profile_id	string	id of the forensic profile associated with the intent

Returns a forensic intent object.

Update a Forensic Intent

```
PUT /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Parameters:

Name	Type	Description
intent_id	string	id of the intent
name	string	name of the intent
inventory_filter_id	string	id of the inventory filter associated with the intent
forensic_config_profile_id	string	id of the forensic profile associated with the intent

Returns a forensic intent object.

Delete a Forensic Intent

```
DELETE /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Parameters:

Name	Type	Description
intent_id	string	id of the intent

Returns a 200 on success.

Forensics Intent Orders

The software agents APIs are associated with managing forensic intent orders.

Forensic profiles are applied to agents using intent. The intents use inventory filters to define the groups of agents. If the filters overlap we need know which to apply. We use the order to define intent priority.

These set of APIs require the `'sensor_management'` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

- Forensic Intent Order Object, on page 103
- Retrieve the Current Forensic Intent Order, on page 103
- Creating a Forensic Intent, on page 102

Forensic Intent Order Object

Attribute	Type	Description
version	string	version of the current ordering
intents	array	name of the intent objects in order
intent_ids	array	array of forensic intent ids

Retrieve the Current Forensic Intent Order

This endpoint returns the current forensic intent order.

```
GET /openapi/v1/inventory_config/forensic_orders
```

Parameters: None

This endpoint returns the current forensic intent order object.

Creating a Forensic Intent Order

```
POST /openapi/v1/inventory_config/forensic_orders
```

Parameters:

Name	Type	Description
version	string	must match the current order
intent_ids	array	array of the intent ids

Returns a forensic intent order object.

Forensics Profiles

The software agents APIs are associated with managing forensic profiles.

Forensic profiles are collections of rules that can be applied to groups of agents using forensic intents.

These set of APIs require the `'sensor_management'` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

Forensic Profile Object

Attribute	Type	Description
id	string	unique identifier of the profile
name	string	name of the profile
forensic_rules	array	array of the rules associated with the profile
created_at	integer	Unix timestamp of when the profile was created
updated_at	integer	Unix timestamp of when the profile was last updated
is_readonly	boolean	indicates if the profile is read only
root_app_scope_id	string	id of the root scope to which the profile belongs

Listing Forensic Profiles

This endpoint lists all existing forensic profiles

```
GET /openapi/v1/inventory_config/forensic_profiles
```

Parameters: None

This endpoint returns an array of forensic profile object summaries.

Retrieving a Single Forensic Profile

```
GET /openapi/v1/inventory_config/forensic_profiles/{profile_id}
```

Parameters:

Name	Type	Description
profile_id	string	id of the profile

Returns a detailed representation of the forensic profile object.

Creating a Forensic Profile

```
POST /openapi/v1/inventory_config/forensic_profiles
```

Parameters:

Name	Type	Description
name	string	name of the profile
root_app_scope_id	string	id of the root scope to which the profile belongs
forensic_rule_ids	array	array of forensic rule ids to be associated with this profile

Returns a forensic profile object.

Update a Forensic Profile

```
PUT /openapi/v1/inventory_config/forensic_profiles/{id}
```

Parameters:

Name	Type	Description
profile_id	string	id of the profile
name	string	name of the profile
forensic_rule_ids	array	array of forensic rule ids to be associated with this profile

Returns a forensic profile object.

Delete a Forensic Profile

```
DELETE /openapi/v1/inventory_config/forensic_profiles/{profile_id}
```

Parameters:

Name	Type	Description
profile_id	string	id of the profile

Returns a 200 on success.

Forensics Rules

The software agents APIs are associated with managing forensic rules.

Forensic rules are used in forensic profiles which are then applied to groups of agents.

These set of APIs require the `'sensor_management'` capability associated with the API key.

Note These APIs are only available to site admins and owners of root scopes.

Forensic Rule Object

Attribute	Type	Description
id	string	unique identifier of the rule
name	string	name of the rule
description	string	description of the rule
type	string	PREDEFINED or USER_DEFINED
eval_group_type	string	AS_INDIVIDUAL or AS_GROUP

Attribute	Type	Description
severity	string	one of IMMEDIATE_ACTION, CRITICAL, HIGH, MEDIUM, LOW
actions	array	array or ALERT or REPORT strings
created_at	integer	Unix timestamp of when the rule was created
updated_at	integer	Unix timestamp of when the rule was last updated

Listing a Forensic Rules

This endpoint lists all existing forensic rules

```
GET /openapi/v1/inventory_config/forensic_rules
```

Parameters: None

This endpoint returns an array of forensic rule object summaries.

Retrieving a Single Forensic Rule

```
GET /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Parameters:

Name	Type	Description
rule_id	string	id of the rule

Returns a detailed representation of the forensic rule object.

Creating a Forensic Rule

```
POST /openapi/v1/inventory_config/forensic_rules
```

Parameters:

Name	Type	Description
root_app_scope_id	string	id of the root scope to which this rule belongs
name	string	name of the rule
description	string	description of the rule
eval_group_type	string	type of the rule

Name	Type	Description
severity	string	severity of the rule
actions	array	array or ALERT or REPORT strings
clause	string	the query clause of the rule.

Returns a forensic rule object.

Update a Forensic Rule

```
PUT /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Parameters:

Name	Type	Description
rule_id	string	id of the rule
name	string	name of the rule
description	string	description of the rule
eval_group_type	string	type of the rule
severity	string	severity of the rule
actions	array	array or ALERT or REPORT strings
clause	string	the query clause of the rule.

Returns a forensic rule object.

Delete a Forensic Rule

```
DELETE /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Parameters:

Name	Type	Description
rule_id	string	id of the rule

Returns a 200 on success.

Enforcement

Policy enforcement is the feature where generated policies are pushed to the assets in the scope associated with a workspace and new firewall rules are written. This set of APIs require the `app_policy_management` capability associated with the API key.

For more information, see Enforce Policies.

Agent Network Policy Config

This endpoint returns an Agent object according to the agent ID. It is useful for fetching the network policy, agent configuration, its version, etc.

```
GET /openapi/v1/enforcement/agents/{aid}/network_policy_config
```

Parameters:

The request URL contains the following parameters

Name	Type	Description
aid	string	Agent UUID for network policy config.

The JSON query body contains the following keys

Name	Type	Description
include_filter_names	boolean	Includes filter names and ID's in network policies.
inject_versions	boolean	Includes ADM workspace versions in network policies.

Response

The response of this endpoint is an Agent object.

Concrete Policy Statistics

This endpoint returns statistics for concrete policies given the agent ID and the concrete policy ID. The endpoint returns an array of Timeseries Concrete Policy Result objects.

```
GET /openapi/v1/enforcement/agents/{aid}/concrete_policies/{cid}/stats?t0=<t0>&t1=<t1>
↪&td=<td>
```

Parameters:

The request URL contains the following parameters

Name	Type	Description
aid	string	Agent UUID for statistics.

Name	Type	Description
cid	string	Concrete Policy UUID for statistics.

The JSON query body contains the following keys

Table 9:

Name	Type	Description
t0	integer	Start time for statistics in epoch time
t1	integer	End time for statistics in epoch time
td	integer or string	Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as “minute”, “hour”, and “day”.

JSON Object Definitions

Agent

Attribute	Type	Description
agent_uuid	string	Agent UUID.
agent_config	object	Agent Config
agent_config_status	object	Agent Config Status
desired_network_policy_config	object	Network Policy Configuration
provisioned_network_policy_config	object	Provisioned Network Policy Config
provisioned_state_update_timestamp	integer	epoch timestamp in seconds when agent acknowledged the above provisioned policy.
desired_policy_update_timestamp	integer	epoch timestamp in seconds when desired_network_policy_config is generated.
agent_info	object	Agent Info
skipped	boolean	true, when concrete policy generation is skipped.

Attribute	Type	Description
message	string	Reason why concrete policy generation is skipped.

Agent Config

Attribute	Type	Description
agent_uuid	string	Agent UUID.
enforcement_enabled	boolean	Config stating is enforcement is enabled on Agent.
fail_mode	string	Fail Mode.
version	number	Agent config version number.
control_tet_rules_only	boolean	Control tet rules only config.
allow_broadcast	boolean	Allow Broadcast config.
allow_multicast	boolean	Allow Multicast config.
allow_link_local	boolean	Allow Link Local config.
enforcement_cpu_quota_mode	string	Enforcement Agent CPU quota mode.
enforcement_cpu_quota_us	string	Enforcement Agent CPU quota micros sec.
enforcement_max_rss_limit	number	Enforcement Agent Max RSS limit.

Network Policy Configuration

Attribute	Type	Description
version	string	Version number.
network_policy	array	Array of Network Policy objects.
address_sets	array	Array of Address Set objects for IP set feature.
container_network_policy	array	Array of ContainerNetworkPolicy objects.

Network Policy

Attribute	Type	Description
priority	string	Priority of concrete policy.

Attribute	Type	Description
enforcement_intent_id	string	Enforcement Intent ID.
concrete_policy_id	string	Concrete Policy ID.
match	object	Match criteria for policy. This field is deprecated.
action	object	Action for policy match.
workspace_id	string	ID for a workspace.
adm_data_set_id	string	Automatic policy discovery data set id of workspace.
adm_data_set_version	string	Automatic policy discovery data set version of the workspace. Set only when inject_versions=true is passed in params.
cluster_edge_id	string	Cluster Edge ID.
policy_intent_group_id	string	Policy intent group ID.
match_set	object	Match Set object for IP set support. Exactly one of match or match_set will be present.
src_filter_id	string	Source inventory filter ID. This will be set when include_filter_names=true passed as params.
src_filter_name	string	Source inventory filter name. This will be set when include_filter_names=true passed as params.
dst_filter_id	string	Destination inventory filter ID. This will be set when include_filter_names=true passed as params.
dst_filter_name	string	Destination Inventory filter name. This will be set when include_filter_names=true passed as params.

ContainerNetworkPolicy

Attribute	Type	Description
pod_id	string	POD ID.

Attribute	Type	Description
network_policy	array	Array of Network Policy objects.
deployment	string	Deployment Name.
service_endpoint	array	List of service endpoint names.

Match

Attribute	Type	Description
src_addr	object	Subnet object for source address.
dst_addr	object	Subnet object for destination address.
src_port_range_start	int	Source port range start.
src_port_range_end	int	Source port range end.
dst_port_range_start	int	Destination port range start.
dst_port_range_end	int	Destination port range end.
ip_protocol	string	IP Protocol.
address_family	string	IPv4 or IPv6 address family.
direction	string	Direction of match, INGRESS or EGRESS.
src_addr_range	object	Address Range object for source address.
dst_add_range	object	Address Range object for destination address.

Action

Attribute	Type	Description
type	string	Action type.

Match Set

Attribute	Type	Description
src_set_id	string	Source set ID of Address Set object in the Network Policy Configuration `address_sets` array.

Attribute	Type	Description
dst_set_id	string	Destination set ID of Address Set object in the Network Policy Configuration `address_sets` array.
src_ports	array	Array of Port Range objects for source ports.
dst_ports	array	Array of Port Range objects for destination ports.
ip_protocol	string	IP Protocol.
address_family	string	IPv4 or IPv6 address family.
direction	string	Direction of match, INGRESS or EGRESS.

Address Set

Attribute	Type	Description
set_id	string	Address set ID.
addr_ranges	array	Array of Address Range objects.
subnets	array	Array of Subnet objects.
addr_family	string	IPv4 or IPv6 address family.

Subnet

Attribute	Type	Description
ip_addr	string	IP address.
prefix_length	int	Prefix length for subnet.

Address Range

Attribute	Type	Description
start_ip_addr	string	Start IP address for range.
end_ip_addr	string	End IP address for range.

Port Range

Attribute	Type	Description
start_port	int	Start port for range.
end_port	int	End port for range.

Agent Config Status

Attribute	Type	Description
disabled	boolean	Config stating is enforcement is disabled on Agent.
current_version	number	Current Agent config version applied on Agent.
highest_seen_version	number	Highest version of agent config received by Agent.

Provisioned Network Policy Config

Attribute	Type	Description
version	string	Network policy config version provisioned by Agent.
error_reason	string	CONFIG_SUCCESS when Agent successfully applied policies else error reason.
disabled	boolean	Config stating is enforcement is disabled on Agent.
current_version	number	Current NPC version applied on Agent.
highest_seen_version	number	Highest version of NPC received by Agent.
policy_status	object	Every network policy status.

Agent Info

Attribute	Type	Description
agent_info_supported	boolean	Agent capability if agent_info is supported.
ipset_supported	boolean	Agent capability if ipsets are supported.

Concrete Policy Result

Attribute	Type	Description
byte_count	int	Byte count for concrete policy hits.
pkt_count	int	Packet count for concrete policy hits.

Timeseries Concrete Policy Result

Attribute	Type	Description
timestamp	string	Timestamp string for aggregation of results.
result	object	Concrete Policy Result

Client Server configuration

Detecting client and server relationships is central to various features in Secure Workload which is why we recommend using the Software Agent whenever possible as it can report the ground truth. Any telemetry monitoring point in the network cannot guarantee to observe every packet for a given flow - due to a wide range of circumstances, for example: two unidirectional halves of a TCP flow may take unique paths through the network - therefore will always unavoidably affected by a level of error.

Secure Workload attempts to detect and minimize these errors without any user interaction by applying machine learning algorithms to each flow, building a statistical model which provides a judgement when inconsistent telemetry is reported. For the majority of cases, users do not need to worry about this set of APIs. However, in some minority of cases the client server detection algorithm does not get the flow direction correct. Features which rely on flow direction, for example, automatic policy discovery may exhibit undesired behavior like opening unnecessary ports.

A set of APIs are provided that can be used to provide hints about known server ports to Secure Workload algorithms. This set of APIs is available to users with root scope ownership role and requires the `app_policy_management` capability associated with the API key for those users.

There are 2 options for Client Server configuration:

Host Config

Configuration of known server ports that are applicable to a specific subset of IP addresses within a root scope

Add server port configuration

This API can be used to provide hints to Secure Workload algorithms about known server ports for a given root scope. You can provide a list of known TCP/UDP server ports for a set of IP addresses belonging to a root scope to aid Secure Workload algorithms with figuring out client server direction correct in flows.

```
POST /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Unique identifier for the root scope.

Additionally, a text file provided as input to this API contains the endpoint server port configuration in the following format:

Endpoint server port configuration

Attribute	Type	Description
ip_address	string	IP Address (can be ipv4 or ipv6 address). Subnets are not allowed.
tcp_server_ports	List of int	List of known TCP server ports corresponding to the ip_address.
udp_server_ports	List of int	List of known UDP server ports corresponding to the ip_address.

Bulk server port configuration

Attribute	Type	Description
host_config	List of Endpoint server port configuration objects.	List of IP addresses with associated known server ports.

Sample python code

```
# contents of below file:
# {"host_config": [
#    {"ip_address": "1.1.1.1",
#     "tcp_server_ports": [100, 101, 102],
#     "udp_server_ports": [103]
#    },
#    {"ip_address": "1.1.1.2",
#     "tcp_server_ports": [200, 201, 202]
#    }
#  ]
# }

file_path = '/<path_to_file>/server_ports.txt'
root_scope_id = '<root-scope-id>'
restclient.upload(file_path,
                  '/adm/%s/server_ports' % root_scope_id,
                  timeout=200) # seconds
```

Note Above API overwrites the full state of known server port configuration in the backend. If you need to modify anything, they need re-upload the full configuration after modifications.

Get server port configuration

This API returns list of known uploaded server ports for endpoints in a root scope.

```
GET /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Unique identifier for the root scope.

Response object: A list of ref:*ServerPortConfig* objects.

Sample python code

```
root_scope_id = '<root-scope-id>'
restclient.get('/adm/%s/server_ports' % root_scope_id)
```

Delete server port configuration

This API deletes server port configuration for specified root scope.

```
DELETE /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Unique identifier for the root scope.

Response object: None.

Sample python code

```
root_scope_id = '<root-scope-id>'
restclient.delete('/adm/%s/server_ports' % root_scope_id)
```

Port Config

Configuration of known server ports that are applicable to all IP addresses that belong to a root scope

Push server port configuration

This API can be used to provide hints to Secure Workload algorithms about known server ports for a given root scope. Users can provide a list of known TCP/UDP server ports for a given root scope to aid Secure Workload algorithms with figuring out client server direction correct in flows. Users also have the option of specifying a service name associated with each server port.

There is also a default list of known services that are applicable to all root scopes(hereafter referred to as global services). This list can be overridden at any point by the user.

Service configuration

A service is defined to be a (port, name) pair.

Attribute	Type	Description
port	int	TCP/UDP server port number
name	string	Service name associated with this port (optional)
override_in_conflicts	boolean	Force host to be provider in case of a conflict (optional)

Bulk service configuration

Attribute	Sub-Attribute	Type	Description
server_ports_config	tcp_service_list	List of *Service configuration* objects.	List of known TCP services
	udp_service_list	List of *Service configuration*objects.	List of known UDP services

```
Push services per root scope:

POST /openapi/v1/adm/{root_scope_id}/server_ports_config
```

Sample python code

```
# contents of below file:
#{"server_ports_config":
#    {
#        "tcp_service_list": [
#                {
#                        "port": 80,
#                        "name" : "http"
#
#                },
#                {
#                        "port": 53,
#                        "name" : "dns"
#                },
#                {
#                        "port": 514,
#                        "name" : "syslog",
#                        "override_in_conflicts": true
#                }
#        ],
#        "udp_service_list": [
#                {
#                        "port": 161
#                },
#                {
#                        "port": 53,
#                        "name": "dns"
#                }
#        ]
#    }
#}

file_path = '/<path_to_file>/server_ports.json'
```

```
# Updating service list for a given root scope
#restclient.upload(file_path,
#                  '/openapi/v1/adm/{root_scope_id}/server_ports_config',
#                  timeout=200) # seconds
```

Note Above API overwrites the full state of known server port configuration in the backend. If user needs to modify anything, they need re-upload the full configuration after modifications.

Retrieve server port configuration

This API returns list of known server ports in a root scope uploaded by the user. Response is *Bulk service configuration.*

```
Retrieve configured services per root scope:
    GET /openapi/v1/adm/{root_scope_id}/server_ports_config

Retrieve configured global services:
    GET /openapi/v1/adm/server_ports_config
```

Remove server port configuration

This API deletes server port configuration for specified root scope.

```
Remove configured services per root scope:
    DELETE /openapi/v1/adm/{root_scope_id}/server_ports_config
```

Software Agents

Agent APIs

The software agents APIs are associated with managing Secure Workload software agents. These set of APIs require the `sensor_management` capability associated with the API key. *GET* APIs below are also available with `flow_inventory_query` capability associated with the API key.

Get software agents

This endpoint returns a list of software agents. Each software agent has two fields to describe its agent type, `agent_type_str` is in plain text while `agent_type` is an enum.

`GET /openapi/v1/sensors`

Parameters:

Name	Type	Description
limit	integer	Limits the number of results returned (optional)

Name	Type	Description
offset	string	Offset is used for paginated requests. If response returns offset then subsequent re- quest must use the same offset to get more results in the next page. (optional)

Get specific software agent

This endpoint returns attributes for the software agent whose UUID is part of the URI. Each software agent has two fields to describe its agent type, <agent_type_str> is in plain text while <agent_type> is an enum.

```
GET /openapi/v1/sensors/{uuid}
```

Deleting software agent

This endpoint is used to decommission a software agent given its UUID.

Use the API with caution; once an agent is deleted, it is no longer available in the Secure Workload dashboard and if the agent is active, flow exports from the agent are not allowed in Secure Workload.

```
DELETE /openapi/v1/sensors/{uuid}
```

Software agent configuration using Intents

This API workflow uses few REST endpoints defined below.

Creating an inventory filter

This endpoint is used to specify criteria that match agent hosts on which user wants to configure software agents.

```
POST /openapi/v1/filters/inventories
```

Parameters:

Name	Type	Description
app_scope_id	string	The scope ID to assign to the inventory filter.
name	string	A name for the inventory filter.
query	json	Filter or match criteria for agent host.

Sample python code

```
# app_scope_id can be retrieved by /app_scopes API
req_payload = {
   "app_scope_id": <app_scope_id>,
   "name": "sensor_config_inventory_filter",
   "query": {
      "type": "eq",
```

```
        "field": "ip",
        "value": <sensor_interface_ip>
    }
}
resp = restclient.post('/filters/inventories',
                json_body=json.dumps(req_payload)))
print resp.status_code
# returned response will contain the created filter and it's ID.
```

Creating a software agent configuration profile

This endpoint is used to specify the set of configuration options to apply to target set of software agents.

```
POST /openapi/v1/inventory_config/profiles
```

Following configuration options can be specified as part of agent configuration profile:

- allow_broadcast: option to allow/disallow broadcast traffic (default value of this option is True).
- allow_multicast: option to allow/disallow multicast traffic (default value of this option is True).
- allow_link_local: option to allow/disallow link local traffic (default value of this option is True).
- auto_upgrade_opt_out: if true, agents are not auto-upgraded during upgrade of Secure Workload cluster.
- data_plane_disabled: if true, agent stops reporting flows to Secure Workload.
- enable_forensics: option to enable collection of forensic events on the workload (agent uses more CPU as a result).
- enable_meltdown: enables Meltdown Exploit detection on the workload (agent uses more CPU as a result).
- enable_pid_lookup: if true, agent tries to attach process information to flows. Note this config option uses more CPU on the end host.
- enforcement_disabled: can be used to disable enforcement on hosts running enforcement agents.
- preserve_existing_rules: option to specify whether to preserve existing iptable rules.
- windows_enforcement_mode: option to use WAF (Windows Advanced Firewall) or WFP (Windows Filtering Platform) (default option is WAF).

For more details about the configuration options, refer to Software Agent Config

Sample python code

```
# Define profile to disable data_plane on agent
req_payload = {
   "root_app_scope_id": <root_app_scope_id>,
   "data_plane_disabled": True,
   "name": "sensor_config_profile_1",
   "enable_pid_lookup": True,
   "enforcement_disabled": False
}
resp = restclient.post('/inventory_config/profiles',
               json_body=json.dumps(req_payload))
print resp.status_code
# returned response will contain the created profile and it's ID.
parsed_resp = json.loads(resp.content)
```

Get software agent configuration profiles

This endpoint returns a list of software agent configuration profiles visible to the user.

```
GET /openapi/v1/inventory_config/profiles
```

Parameters: None

Get specific software agent configuration profile

This endpoint returns an instance of software agent configuration profile.

```
GET /openapi/v1/inventory_config/profiles/{profile_id}
```

Returns the software agent configuration profile object associated with the specified ID.

Update a software agent configuration profile

This endpoint updates a software agent configuration profile.

```
PUT /openapi/v1/inventory_config/profiles/{profile_id}
```

Following configuration options can be specified as part of agent configuration profile:

- allow_broadcast: option to allow/disallow broadcast traffic (default value of this option is True).
- allow_multicast: option to allow/disallow multicast traffic (default value of this option is True).
- allow_link_local: option to allow/disallow link local traffic (default value of this option is True).
- auto_upgrade_opt_out: if true, agents are not auto-upgraded during upgrade of Secure Workload cluster.
- data_plane_disabled: if true, agent stops reporting flows to Secure Workload.
- enable_forensics: option to enable collection of forensic events on the workload (agent uses more CPU as a result).
- enable_meltdown: enables Meltdown Exploit detection on the workload (agent uses more CPU as a result).
- enable_pid_lookup: if true, agent tries to attach process information to flows. Note this config option uses more CPU on the end host.
- enforcement_disabled: can be used to disable enforcement on hosts running enforcement agents.
- preserve_existing_rules: option to specify whether to preserve existing iptable rules.
- windows_enforcement_mode: option to use WAF (Windows Advanced Firewall) or WFP (Windows Filtering Platform) (default option is WAF).

For more details about the configuration options, refer to Software Agent Config

Returns the modified software agent configuration profile object associate with the specified ID.

Delete a software agent configuration profile

This endpoint deletes the specified software agent configuration profile.

```
DELETE /openapi/v1/inventory_config/profiles/{profile_id}
```

Creating a software agent configuration intent

This endpoint is used to specify the intent to apply set of configuration options to specified set of software agents. This will create the intent and updates the intent order by adding the newly created intent to the order.

```
POST /openapi/v1/inventory_config/intents
```

Sample python code

```
req_payload = {
  "inventory_config_profile_id": <>,
  "inventory_filter_id": <>
}
resp = restclient.post('/inventory_config/intents',
               json_body=json.dumps(req_payload))
print resp.status_code
# returned response will contain the created intent object and it's ID.
```

Specifying order of intents

This endpoint is used to specify the ordering of various software agent configuration intents. For example, there could be two intents – one to enable process ID lookup on development machines and second one to disable process ID lookup on windows machines. If the first intent has higher priority, then development windows machines will have process ID lookup enabled. NOTE: By default, when intent is created, it is added to the beginning of intent orders list. This endpoint is only to be used if end user needs to modify the existing order of intents.

```
POST /openapi/v1/inventory_config/orders
```

Sample python code

```
# Read the agent config intents ordered list
resp = restclient.get('/inventory_config/orders')
order_result_json = json.loads(resp.content)

# Modify the list by prepending the new intent in the list
order_rslt_json['intent_ids'].insert(0,<intent_id>)

# Post the new ordering back to the server
resp = restclient.post('/inventory_config/orders',
               json_body=json.dumps(order_rslt_json))
```

Remove agent config intent

This endpoint is used to remove a specific agent configuration intent.

```
DELETE /openapi/v1/inventory_config/intents/{intent_id}
```

Sample python code

```
intent_id = '588a51dcb5b30d0ee6da084a'
resp = restclient.delete('/inventory_config/intents/%s' % intent_id)
```

Interface Config Intents

The recommended way to assign VRFs to agents is using Remote VRF configuration settings. In rare cases, when agent hosts may have multiple interfaces that need to be assigned different VRFs, users can choose to assign them VRFs using Interface Config Intents. Go to **Manage** > **Agents** and click the **Configure** tab.

Inventory Config Intent Object

The GET and POST methods return an array of inventory config intent JSON objects. The object's attributes are described below:

Attribute	Type	Description
vrf_id	integer	VRF ID integer
vrf_name	string	VRF Name
inventory_filter_id	string	Inventory Filter ID
inventory_filter	JSON	Inventory filter. See OpenAPI > Inventory Filters for more details.

Get Interface Config Intents

This endpoint returns a list of inventory config intents to the user.

```
GET /openapi/v1/inventory_config/interface_intents
```

Parameters: None

Create or Update list of Interface Config Intents

This endpoint is used to create or modify list of interface config intents. The API takes an ordered list of intents. To remove an intent in this list, users would need to read the existing list of intents, modify the list and write the modified list back.

```
POST /openapi/v1/inventory_config/interface_intents
```

Parameters:

Name	Type	Description
inventory_filter_id	string	Inventory filter ID to match interface
vrf_id	integer	VRF ID to assign interface

Sample python code

```
  req_payload = {
    "intents": [
      {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_1>},
      {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_2>}
    ]
  }
  resp = restclient.post('/inventory_config/interface_intents',
json_body=json.dumps(req_payload))
```

VRF configuration for agents behind NAT

Following set of APIs are useful to specify policies to assign VRFs to agents behind NAT boxes. These set of APIs require the `sensor_management` capability associated with the API key and are only available to site admin users.

List VRF configuration rules for agents behind NAT

This endpoint returns a list of VRF configuration rules applicable to agents behind NAT.

```
GET /openapi/v1/agentnatconfig
```

Create a new VRF configuration applicable to agents behind NAT

This endpoint is used to specify criteria for VRF labeling for hosts based on their source IP and source port as seen by Secure Workload appliance.

```
POST /openapi/v1/agentnatconfig
```

Parameters:

Name	Type	Description
src_subnet	string	Subnet to which source IP can belong to (CIDR notation).
src_port_range_start	integer	Lower bound of source port range (0-65535).
src_port_range_end	integer	Upper bound of source port range (0-65535).
vrf_id	integer	VRF ID to use for labeling flows for agents whose source address and port falls in the above specified range.

Sample python code

```
req_payload = {
    src_subnet: 10.1.1.0/24,          # src IP range for sensors
    src_port_range_start: 0,
    src_port_range_end: 65535,
    vrf_id: 676767                    # VRF ID to assign
}

resp = rc.post('/agentnatconfig', json_body=json.dumps(req_payload))
print resp.status_code
```

Delete existing VRF configuration

```
DELETE /openapi/v1/agentnatconfig/{nat_config_id}
```

Secure Workload software download

The Secure Workload software download feature provides a way to download software packages for Secure Workload agents. These set of APIs require the `software_download` capability associated with the API key. This capability is only available to site admin users, root scope owners and users with agent installer roles.

API to get supported platforms

This end point returns the list of supported platforms.

```
GET /openapi/v1/sw_assets/platforms
```

Parameters: None

Response object: Returns the list of supported platforms.

Sample python code

The sample code below retrieves all the supported platforms.

```
resp = restclient.get('/sw_assets/platforms')
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
{"results": [{"platform": "OracleServer-6.3", "agent_type": "enforcer", "arch": "x86_64"},
 {"platform": "MSWindows8Enterprise", "agent_type": "legacy_sensor", "arch": "x86_64"}]}
```

API to get supported software version

This endpoint returns the list of supported software version for specified “agent_type”, “package_type”, “platform” and “architecture”.

```
GET
/openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_type=<pkg_type>&arch=<arch>&list_version=<list_version>&installation_id=<installer_id>
```

where <agent_type>, <platform> and <arch> can be any one of the results retrieved from the **API to get supported platforms**, and <pkg_type> can be either “sensor_w_cfg” or “sensor_bin_pkg”. Both <pkg_type> and <agent_type> are optional but at least one of them should be specified. <list_version> must be “True” to enable this API.

Parameters: The request URL contains the following parameters.

Name	Type	Description
platform	string	Specify the platform.
agent_type	string	(optional) Specify the agent type.
pkg_type	string	(optional) Specify the package type, the value can be either “sensor_w_cfg” or “sensor_bin_pkg”.
arch	string	Specify the architecture.

Name	Type	Description
list_version	string	Set to “True” to enable software version search.

Response object: Returns a list of supported software version.

Sample python code

```
resp =
restclient.get('/sw_assets/download?platform=OracleServer-6.3&pkg_type=sensor_w_cfg&arch=x86_64&list_version=True')
if resp.status_code == 200:
   print resp.content
```

```
resp =
[illegible]
if resp.status_code == 200:
   print resp.content
```

Sample response

```
3.3.1.30.devel
3.3.1.31.devel
```

API to create installer ID

This endpoint creates an installer ID for API to download Secure Workload software.

```
GET /openapi/v1/sw_assets/installation_id
```

Response object: Returns an installer ID that can be used in API to download Secure Workload software.

Sample python code

```
resp = restclient.get('/sw_assets/installation_id')
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
"[illegible]"
```

API to download Secure Workload software

This endpoint enables clients to download the software for specified “agent_type”, “package_type”, “platform”, “ar- chitecture” and “sensor_version”.

```
GET
/openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_type=<pkg_type>&arch=<arch>&sensor_version=<sensor_version>&installation_id=<installer_id>
```

where <agent_type>, <platform> and <arch> can be any one of the results retrieved from the **API to get supported platforms**, and <pkg_type> can be either “sensor_w_cfg” or “sensor_bin_pkg”. Both <pkg_type> and <agent_type> are optional but at least one of them should be specified. <sensor_version> can be any one of the results retrieved from the **API to get supported software version**. If “sensor_version” is not specified, it will download the **latest** software.

Parameters: The request URL contains the following parameters.

Name	Type	Description
platform	string	Specify the platform.
agent_type	string	(optional) Specify the agent type.
pkg_type	string	(optional) Specify the package type, the value can be either “sensor_w_cfg” or “sensor_bin_pkg”.
arch	string	Specify the architecture.
sensor_version	string	(optional) Specify the software version, defaults to empty string.

Response object: Returns the Secure Workload software for the given parameters.

Sample python code

```
resp =
if resp.status_code == 200:
   print 'file downloaded successfully'
```

Secure Workload Agents Upgrade

The Secure Workload agents upgrade feature provides a way to upgrade installed Secure Workload agents to specific version. It only updates the metadata, actual upgrade will happen during next check-in. The API requires the `software_download` capability associated with the API key. This capability is only available to site admin users, root scope owners or users with agent installer roles.

API to upgrade an agent to specific version

This end point triggers the agent given its “UUID” upgrade to specific “sensor_version”, the latest version will be applied if “sensor_version” is not provided. This API won’t proceed downgrade requests.

```
POST /openapi/v1/sensors/{UUID}/upgrade?sensor_version=<sensor_version>
```

where <sensor_version> can be any one of the results retrieved from the API to get supported software version .

Parameters: The request URL contains the following parameters

Name	Type	Description
sensor_version	string	(optional) Specify the desired version, the latest version will be applied by default

Returns the status for this upgrade request.

Sample python code

```
resp = restclient.post('/openapi/v1/sensors/{UUID}/upgrade?sensor_version=3.4.1.1.devel')

if resp.status_code == 200:
  print 'agent upgrade was triggered successfully and in progress'
elif resp.status_code == 304:
  print 'provided version is not newer than current version'
elif resp.status_code == 400:
  print 'provided version is invalid'
elif resp.status_code == 403:
  print 'user does not have required capability'
elif resp.status_code == 404:
  print 'agent with {UUID} does not exist'
```

User Uploaded Filehashes

Users can upload a list of filehashes to Secure Workload and specify whether those hashes are benign or flagged. Secure Workload flags processes with the respective binary hashes accordingly.

This set of APIs can be used to upload or remove a list of filehashes to Secure Workload. To call these APIs, use an API key with the `user_data_upload` capability.

Note You can have up to 1 million file hashes per root scope. 500000 for both benign and flagged hashes each.

The following APIs are available to scope owners and site admins and are used to upload/download/remove filehashes in a single root scope on the |product| appliance.

User Filehash Upload

This endpoint is used to upload a CSV file with filehash for a root scope on the Secure Workload appliance. The column headers `HashType` and `FileHash` must appear in the CSV file. `HashType` should be `SHA-1` or `SHA-256`, FileHash must not be empty and must be in the format of 40-hex SHA1 or 64-hex SHA256.

`FileName` and `Notes` headers are optional. Given file name should not exceed maximum length of 150 characters and given notes should not exceed maximum length of 1024 characters.

`POST /openapi/v1/assets/user_filehash/upload/{rootAppScopeNameOrID}/{benignOrflagged}`

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeNameOrID	string	Root scope name or ID.
benignOrflagged	string	Can be one of `benign` or `flagged`.

Response object: None

Sample python code

```
# Sample CSV File
# HashType,FileHash,FileName,Notes
# SHA-1,1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29,application_1.exe,Sample Notes
#
```

```
SHA-256,8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4,application_2.exe,Sample
 Notes

  file_path = '/<path_to_file>/user_filehash.csv'
  root_app_scope_name = 'Tetration'
  restclient.upload(file_path, '/assets/user_filehash/upload/%s/benign' % root_app_scope_name)
```

User Filehash Delete

This endpoint is used to upload a CSV file to delete filehashes from the root scope on the Secure Workload appliance. CSV file must have `FileHash` as a header.

```
POST /openapi/v1/assets/user_filehash/delete/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeNameOrID	string	Root scope name or ID.
benignOrflagged	string	Can be one of `benign` and `flagged`.

Response object: None

Sample python code

```
  # Sample CSV File
  # FileHash
  # 1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29
  # 8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4

  file_path = '/<path_to_file>/user_filehash.csv'
  root_app_scope_name = 'Tetration'
  restclient.upload(file_path, '/assets/user_filehash/delete/' + root_app_scope_name +
'/benign')
```

User Filehash Download

This endpoint returns the user file hash for the given root scope on the Secure Workload appliance as a CSV file. The CSV file has the headers `HashType, FileHash, FileName` and `Notes` in the respective order.

```
GET /openapi/v1/assets/user_filehash/download/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeNameOrID	string	Root scope name or ID.
benignOrflagged	string	Can be one of benign or flagged.

Response object: None

Sample python code

```
  file_path = '/<path_to_file>/output_user_filehash.csv'
```

```
  root_app_scope_name = 'Tetration'
  restclient.download(file_path, '/assets/user_filehash/download/%s/benign' %
root_app_scope_name)
```

User-Defined Labels

These APIs are used to add or remove user-defined labels that label flows and inventory items on the Secure Workload appliance. To call these APIs, use an API key with the `user_data_upload` capability. See the Label schema section of the UI user guide for guidelines governing keys and values that are used for labeling flows and inventory items.

Note Refer to Importing Custom Labels for instructions on accessing this functionality via the UI.

Note Refer to Label Limits for limits on the number of IPv4/IPv6 addresses or subnets that can be uploaded.

Scope-Dependent APIs

The following APIs are used to get/set/delete labels in a root scope on the Secure Workload appliance. They are available to root **scope owners** and **site admins**. Also, the GET API calls are available to users with **read access** to the root scope.

Get Inventory Label

This endpoint returns labels for an IPv4/IPv6 address or subnet in root scope on the Secure Workload appliance. The address/subnet used to query this endpoint must exactly match the one used for uploading labels.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.
IPorSubnet	string	IPv4/IPv6 address or subnet.

Response object:

Name	Type	Description
attributes	JSON	Key/value map for labeling matching flows and inventory items

Sample python code

```
  root_app_scope_name = 'Tetration'
  restclient.get('/inventory/tags/%s' % root_app_scope_name, params={'ip': '10.1.1.1/24'})
```

Search Inventory Label

This endpoint allows for searching labels for an IPv4/IPv6 address or subnet in root scope on the Secure Workload appliance.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}/search?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.
IPorSubnet	string	IPv4/IPv6 address or subnet.

Response object: This API returns a list of objects of the following format

Name	Type	Description
key	string	IPv4/IPv6 address or subnet.
updatedAt	integer	Unix timestamp of when the labels were updated.
value	JSON	Key/value map of attributes for the key.

Sample python code

```
  root_app_scope_name = 'Tetration Scope'
  encoded_root_app_scope_name = urllib.quote(root_app_scope_name, safe='')
  restclient.get('/inventory/tags/%s/search' % encoded_root_app_scope_name, params={'ip':
'10.1.1.1/24'})
```

Set Inventory Label

This endpoint is used to set labels for labeling flows and inventory items in root scope on the Secure Workload appliance.

```
POST /openapi/v1/inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

The JSON query body contains the following keys

Name	Type	Description
ip	string	IPv4/IPv6 address or subnet.
attributes	JSON	Key/value map for labeling matching flows and inventory items

Response object:

Name	Type	Description
warnings	JSON	Key/value map containing warnings that are encountered while setting labels.

Sample python code

```
  root_app_scope_name = 'Tetration'
  req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location': 'CA'}}

  restclient.post('/inventory/tags/%s' % root_app_scope_name,
json_body=json.dumps(req_payload))
```

Delete Inventory Label

This endpoint deletes labels for an IPv4/IPv6 address or subnet in a root scope on the Secure Workload appliance.

```
DELETE /openapi/v1/inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

The JSON query body contains the following keys

Name	Type	Description
ip	string	IPv4/IPv6 address or subnet

Sample python code

```
  root_app_scope_name = 'Tetration'
  req_payload = {'ip': '10.1.1.1/24'}
  restclient.delete('/inventory/tags/%s' % root_app_scope_name,
json_body=json.dumps(req_payload))
```

Upload Labels

This endpoint is used to upload a CSV file with labels for labeling flows and inventory items in a root scope on the Secure Workload appliance. A column header with name `IP` must appear in the CSV file. Of the remaining column headers, up to 32 can be used to annotate flows and inventory items. To use non-English characters in labels, the uploaded csv file must be in UTF-8 format.

```
POST /openapi/v1/assets/cmdb/upload/{rootAppScopeName}
```

Parameters:

User must provide an operation type `(X-Tetration-Oper)` as a parameter to this API. `X-Tetration-Oper` can be one of the following:

- add: Appends labels to new and existing addresses/subnets. Resolves conflicts by selecting new labels over existing ones. For example, if labels for an address in the database are *{“foo”: “1”, “bar”: “2”}*, and the CSV file contains *{“z”: “1”, “bar”: “3”}*, *add* sets labels for this address to *{“foo”: “1”, “z”: “1”, “bar”: “3”}*.
- overwrite: inserts labels for new addresses/subnets and replaces labels for existing ones. For example, if labels for an address in the database are *{“foo”: “1”, “bar”: “2”}* and the CSV file contains *{“z”: “1”, “bar”: “3”}*, *overwrite* sets labels for this address to *{“z”: “1”, “bar”: “3”}*.
- merge: Merges labels to existing addresses/subnets. Resolves conflicts by selecting nonempty values over empty values. For example, if labels for an address in the database are *{“foo”: “1”, “bar”: “2”, “qux”: “”, “corge”: “4”}*, and the CSV file contains *{“z”: “1”, “bar”: “”, “qux”: “3”, “corge”: “4-updated”}*, *merge* sets that are labels for this address to *{“foo”: “1”, “z”: “1”, “bar”: “2”, “qux”: “3”, “corge”: “4-updated”}*.

Note Value of “bar” in not reset to “”(empty), instead existing value of “bar”=”2” is preserved.

- delete: removes labels for an address/subnet.

Response object:

Name	Type	Description
warnings	JSON	Key/value map containing warnings that are encountered while setting labels.

Sample python code

```
file_path = '/<path_to_file>/user_annotations.csv'
root_app_scope_name = 'Tetration'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload/%s' % root_app_scope_name, req_payload)
```

Upload Labels Using JSON Format

This endpoint is used to upload labels for labeling flows and inventory items on the Secure Workload appliance using JSON format. Attributes in ip_tags must be a subset of headers, up to 32 headers can be used to annotate flows and inventory items. To use non-English characters in labels, the json payload must be in UTF-8 format.

```
POST /openapi/v1/multi_inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameter

Name	Type	Description
rootAppScopeName	string	Root scope name

Table 10: Payload Parameters

Name	Type	Description
headers	array	Array of strings that specifies the label names
operation	string	Either **add** or **merge**, if not specified it is considered as **add**.
ip_tags	array	Array of ip_tags object

Table 11: ip_tags

Name	Type	Description
ip	string	Valid IPV4/IPV6 Address or Subnet
attributes	JSON	JSON of label, value string pairs; labels must be a subset of headers.

Note 1. If record labels are not a subset of given headers, a warning is generated to alert the user of a discrepancy between the given headers and the labels in the record.

2. This endpoint will only allow user to upload a maximum of 95,000 entries and 36 attributes.

- For add request, *operation* is optional, if not specified, it is considered as **add**.
- For merge request, *operation* must be specified as **merge**.

3. You can upload a maximum of 95,000 entries and 36 attributes.

Sample python code for add request

```
  {
    "headers" : ["key1", "key2"],
    "operation": "add"
    "ip_tags" : [
        {
            "ip": "10.10.10.11",
            "attributes": {
                "key1": "val1",
                "key2": "val2"
            }
        },
        {
            "ip": "10.10.10.12",
            "attributes": {
                "key1": "val111",
                "key2": "val2"
            }
        }
    ]
}
```

Sample python code for merge request

```
  {{
    "headers" : ["key1", "key2"],
    "operation": "merge"
    "ip_tags" : [
        {
            "ip": "10.10.10.11",
            "attributes": {
                "key1": "val1",
                "key2": "val2"
            }
        },
        {
            "ip": "10.10.10.12",
            "attributes": {
                "key1": "val1",
                "key2": "val2"
            }
        }
    ]
}
resp = restclient.post('/multi_inventory/tags/%s' % rootAppScopeName,
json_body=json.dumps(req_payload))
```

Download User Labels

This endpoint returns user-uploaded labels for a root scope on the Secure Workload appliance as a CSV file.

```
GET /openapi/v1/assets/cmdb/download/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response:

Content-Type: *text/csv*

CSV file containing user uploaded labels for the scope.

Sample python code

```
  file_path = '/<path_to_file>/output.csv'
  root_app_scope_name = 'Tetration'
  restclient.download(file_path, '/assets/cmdb/download/%s' % root_app_scope_name)
```

Get Column Headers

This endpoint returns a list of column headers for a root scope on the Secure Workload appliance.

```
GET /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: An array of facets available for a label.

Sample python code

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/attributenames/%s' % root_app_scope_name)
```

Delete Column Header

This endpoint deletes a column header in a root scope on the Secure Workload appliance. Deleting a column header drops it from the list of labeled facets and removes it from existing labels.

```
DELETE /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}/{attributeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.
attributeName	string	Attribute being deleted.

Response object: None

Sample python code

```
root_app_scope_name = 'Tetration'
attribute_name = 'column1'
resp = restclient.delete('/assets/cmdb/attributenames/%s/%s' % (root_app_scope_name,
attribute_name))
```

Delete Labels Using JSON Format

This endpoint deletes labels for multiple IPv4/IPv6 address or subnet using JSON format.

```
DELETE /openapi/v1/multi_inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name

Table 12: Payload Parameters

Name	Type	Description
headers	array	(Optional) array of strings that specifies the label names
ip_tags	array	Array of ip_tags object

Table 13: ip_tags

Name	Type	Description
ip	string	Valid IPV4/IPV6 Address or Subnet
attributes	JSON	(Optional) JSON of label, value string pairs; labels must be a subset of headers.

Sample python code

```
  {
    "ip_tags" : [
        {
            "ip": "10.10.10.11",
          },
        {
            "ip": "10.10.10.12",
            "attributes": {
                "key1": "val1",
                "key2": "val2"
            }
        }
    ]
}
resp = restclient.delete('/multi_inventory/tags/%s' % rootAppScopeName,
json_body=json.dumps(req_payload))
```

Get List of Labeled Facets

This endpoint returns a list of labeled facets for a root scope on the Secure Workload appliance. Labeled facets are a subset of column headers in the uploaded CSV file that is used for annotating flows and inventory items in that scope.

```
GET /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: An array of labeled facets for the root scope.

Sample python code

```
   root_app_scope_name = 'Tetration'
   resp = restclient.get('/assets/cmdb/annotations/%s' % root_app_scope_name)
```

Update List of Labeled Facets

This endpoint updates list of facets that are used for annotating flows and inventory items in a root scope on the Secure Workload appliance.

```
PUT /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: None

Sample python code

```
# the following list is a subset of column headers in the
# uploaded CSV file
req_payload = ['location', 'region', 'detail']
root_app_scope_name = 'Tetration'
restclient.put('/assets/cmdb/annotations/%s' % root_app_scope_name,
               json_body=json.dumps(req_payload))
```

Flush User Uploaded Labels

This endpoint flushes labels for flows and inventory items in root scope on the Secure Workload appliance. The changes affect new data; older labeled data remains unaltered.

```
POST /openapi/v1/assets/cmdb/flush/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: None

Sample python code

```
root_app_scope_name = 'Tetration'
restclient.post('/assets/cmdb/flush/%s' % root_app_scope_name)
```

Scope-Independent APIs

The following APIs can span multiple scopes on the Secure Workload appliance.

Note The number of scope independent and dependent annotated facets must not exceed 32 for any root scope.

Upload Labels

This endpoint is used to upload a CSV file with labels for labeling flows and inventory items on the Secure Workload appliance. Column headers with names `IP` and `VRF` must appear in the CSV file and `VRF` must match the root scope for a label. Of the remaining column headers, up to 32 can be used to annotate flows and inventory items.

```
POST /openapi/v1/assets/cmdb/upload
```

Parameters:

User must provide an operation type `(X-Tetration-Oper)` as a parameter to this API to specify the operation to be performed.

Response object:

Name	Type	Description
warnings	JSON	Key or value map containing warnings that are encountered while setting labels.

Sample python code

```
file_path = '/<path_to_file>/user_annotations.csv'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload', req_payload)
```

Download User Labels

This endpoint returns the user-uploaded labels for all scopes on the Secure Workload appliance as a CSV file.

```
GET /openapi/v1/assets/cmdb/download
```

Parameters: None

Response:

Content-Type: *text/csv*

CSV file containing user uploaded labels for the scope.

Sample python code

```
file_path = '/<path_to_file>/output.csv'
restclient.download(file_path, '/assets/cmdb/download')
```

Scope-Independent Labels

These labels are not tied to a particular root scope and apply to all scopes on the appliance.

Get Inventory Label

These endpoints return scope independent labels for an IPv4/IPv6 address or subnet on the Secure Workload appliance. The address or subnet that is used to query this endpoint must exactly match the one used for uploading labels.

```
GET /openapi/v1/si_inventory/tags?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters.

Name	Type	Description
IPorSubnet	string	IPv4/IPv6 address or subnet.

Response object:

Name	Type	Description
attributes	JSON	Key or value map for labeling matching flows and inventory items

Sample python code

```
restclient.get('/si_inventory/tags', params={'ip': '10.1.1.1/24'})
```

Search Inventory Label

This endpoint allows for searching labels for an IPv4/IPv6 address or subnet on the Secure Workload appliance.

```
GET /openapi/v1/si_inventory/tags/search?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
IPorSubnet	string	IPv4/IPv6 address or subnet.

Response object: This API returns a list of objects of the following format

Name	Type	Description
key	string	IPv4/IPv6 address or subnet.
updatedAt	integer	Unix timestamp of when the labels were updated.
value	JSON	Key or value map of attributes for the key.

Sample python code

```
restclient.get('/si_inventory/tags/search', params={'ip': '10.1.1.1/24'})
```

Set Inventory Label

This endpoint is used to set labels for labeling flows and inventory items on the Secure Workload appliance.

```
POST /openapi/v1/si_inventory/tags
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
ip	string	IPv4/IPv6 address or subnet.
attributes	JSON	Key or value map for labeling matching flows and inventory items.

Response object:

Name	Type	Description
warnings	JSON	Key or value map containing warnings that are encountered while setting labels.

Sample python code

```
req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location': 'CA'}}

restclient.post('/si_inventory/tags', json_body=json.dumps(req_payload))
```

Delete Inventory Label

This endpoint deletes labels for an IPv4/IPv6 address or subnet on the Secure Workload appliance.

```
DELETE /openapi/v1/si_inventory/tags
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
ip	string	IPv4/IPv6 address or subnet

Sample python code

```
req_payload = {'ip': '10.1.1.1/24'}
restclient.delete('/si_inventory/tags, json_body=json.dumps(req_payload))
```

Get List of Labeled Facets

This endpoint returns a list of scope-independent labeled facets on the Secure Workload appliance. Labeled facets are a subset of column headers that are used for annotating flows and inventory items across all scopes.

Note Exclude the scope name from the request url to view and update the list of annotated scope-independent facets.

```
GET /openapi/v1/assets/cmdb/annotations
```

Response object: An array of scope-independent labeled facets.

Sample python code

```
resp = restclient.get('/assets/cmdb/annotations')
```

Update List of Labeled Facets

This endpoint updates list of scope-independent facets that are used for annotating flows and inventory items on the Secure Workload appliance.

```
PUT /openapi/v1/assets/cmdb/annotations
```

Response object: None

Sample python code

```
# the following list is a subset of column headers in the
# uploaded CSV file
req_payload = ['location', 'region', 'detail']
restclient.put('/assets/cmdb/annotations',
               json_body=json.dumps(req_payload))
```

Virtual Routing and Forwarding

This set of APIs manages Virtual Routing and Forwarding (VRF).

Note These APIs are only available to site admins.

VRF Object

The VRF object attributes are described below:

Attribute	Type	Description
id	int	Unique identifier for the VRF.
name	string	User specified name of the VRF.
tenant_id	int	ID of parent tenant.
root_app_scope_id	string	ID of associated root scope.
created_at	integer	Unix timestamp when the VRF was created.
updated_at	integer	Unix timestamp when the VRF was last updated.

Get VRFs

This endpoint returns a list of VRFs. This API is available to API keys with `sensor_management or flow_inventory_query` capability.

```
GET /openapi/v1/vrfs
```

Parameters: None

Response object: Returns a list of VRF objects.

Sample python code

```
resp = restclient.get('/vrfs')
```

Create a VRF

This endpoint is used to create new VRFs. An associated root scope will automatically be created with a query matching the VRF ID. This API is available to API keys with `sensor_management` capability.

```
POST /openapi/v1/vrfs
```

Parameters:

Name	Type	Description
id	int	(optional) Unique identifier for the VRF. If unspecified, Secure Workload cluster will generate a unique ID for the newly created VRF. Best practice is to let Secure Workload generate these IDs instead of caller explicitly specifying unique IDs.
tenant_id	int	(optional) ID of parent tenant.
name	string	User specified name of the VRF.
apply_monitoring_rules	boolean	(optional) Whether collection rules should be applied for the VRF. Defaults to 'false'.

The `tenant_id` is optional. If not provided, the VRF will be added to the tenant with the same id as the VRF, autocreating if necessary. If the `tenant_id` is provided, the tenant will not be automatically created, and an error will be returned if the tenant does not exist.

Response object: Returns the newly created VRF object.

Sample python code

```
req_payload = {
    "tenant_id": <tenant_id>,
    "name": "Test",
    "apply_monitoring_rules": True
}
resp = restclient.post('/vrfs', json_body=json.dumps(req_payload))
```

Get Specific VRF

This endpoint returns information for the specified VRF ID. This API is available to API keys with `sensor_management or flow_inventory_query` capability.

```
GET /openapi/v1/vrfs/{vrf_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
vrf_id	int	Unique identifier for the VRF.

Response object: Returns a VRF object that is associated with the specified ID.

Sample python code

```
vrf_id = 676767
resp = restclient.get('/vrfs/%d'% vrf_id)
```

Update a VRF

This endpoint updates a VRF. This API is available to API keys with `sensor_management` capability.

```
PUT /openapi/v1/vrfs/{vrf_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
vrf_id	int	Unique identifier for the VRF.

The JSON request body contains the following parameters

Name	Type	Description
name	string	User specified name of the VRF.
apply_monitoring_rules	boolean	(optional) Whether collection rules should be applied to the VRF.

Response object: Returns the modified VRF object that is associated with the specified ID.

Sample python code

```
vrf_id = 676767
req_payload = {
    "name": "Test",
    "apply_monitoring_rules": True
}
resp = restclient.put('/vrfs/%d'% vrf_id,
                      json_body=json.dumps(req_payload))
```

Delete Specific VRF

This endpoint deletes a VRF. It fails if there is associated root scope. This API is available to API keys with `sensor_management` capability.

```
DELETE /openapi/v1/vrfs/{vrf_id}
```

Parameters: The following parameter is part of the URL.

Name	Type	Description
vrf_id	int	Unique identifier for the VRF.

Sample python code

```
vrf_id = 676767
resp = restclient.delete('/vrfs/%d'% vrf_id)
```

Orchestrators

This set of APIs can be used to manage external orchestrator inventory learning in Secure Workload cluster deployment. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator types are ‘vcenter’ (vCenter 6.5 and later), ‘kubernetes’, ‘dns’, ‘f5’, ‘netscaler’, ‘infoblox’ and ‘Cisco FMC’. Supported user interface located at External Orchestrators.

Orchestrator Object

The orchestrator object attributes are described below - some of the fields are applicable only for specific orchestrator types; restrictions are mentioned in the table below.

Attribute	Type	Description
id	string	Unique identifier for the orchestrator.
name	string	User specified name of the orchestrator.
type	string	Type of orchestrator - supported values (*vcenter, kubernetes, f5, netscaler, infoblox, dns)*
description	string	User specified description of the orchestrator.
username	string	Username for the orchestration endpoint. (unnecessary for *dns*)
password	string	Password for the orchestration endpoint. (unnecessary for *dns*)
certificate	string	Client certificate used for authentication (unnecessary for *dns*)
key	string	Key corresponding to client certificate (unnecessary for *dns*)
ca_certificate	string	CA Certificate to validate orchestration endpoint (unnecessary for *dns*)
auth_token	string	Opaque authentication token (bearer token) (applies only for *kubernetes*)

Attribute	Type	Description
insecure	boolean	Turn off strict SSL verification
delta_interval	integer	Delta polling interval in seconds Secure Workload Inventory manager will perform polling for incremental changes every delta_interval seconds. Note this parameter is not applicable for Infoblox and Secure Firewall Management Center.
full_snapshot_interval	integer	Full snapshot interval in seconds Secure Workload Inventory manager will perform a full refresh poll from the or- chestrator
verbose_tsdb_metrics	boolean	Per-Endpoint TSDB metrics
hosts_list	Array	Array of { “host_name”, port_number} pairs that specify how Secure Workload must connect to the orchestrator
use_secureconnector_tunnel	boolean	Tunnel connections to this orchestrator’s hosts through the Secure Connector tunnel
route_domain	integer	Route Domain number to poll on F5 LoadBalancers (applies only for *f5*)
dns_zones	Array	Array of strings containing the DNS zones to poll from the DNS server (only for *dns*). Each DNS Zone entry MUST end with a .
enable_enforcement	boolean	Applicable only for external orchestrators with policy enforcement support such as firewalls and load balancers. Examples are *Secure Firewall Management Center*, *F5 BIGIP* and *Citrix Netscaler*. This flag is false (policy en- forcement is disabled) by default. If true, the external or- chestrator will deploy policies to the given load balancer appliance when policy enforcement is performed for the workspace.
ingress_controllers	object	Array of Ingress Controller objects.

Attribute	Type	Description
fmc_enforcement_mode	string	Applicable only for *Secure Firewall Management Center external orchestrator* and must be either *merge* (default) or *override*. The first instance instructs Secure Firewall Management Center policy enforcer to put all Secure Workload policy rules before any existing prefilter rules, while the latter instance will remove all prefilter rules created by the users.
infoblox_config	object	Applicable only for *Infoblox external orchestrator*. Infoblox Config record type selectors.

Ingress Controller

Attribute	Type	Description
pod_selector	object	Pod Selector
controller_config	object	Controller Config

Pod Selector

Attribute	Type	Description
namespace	string	Namespace where the Ingress controller pod is running.
labels	Array	Array of {“key”, “value”} pairs that specify the labels of ingress controller pods.

Controller Config

Attribute	Type	Description
ingress_class	string	Name of the ingress class which ingress controller satisfies.
namespace	string	Namespace is the name of the namespace which ingress controller satisfies.
http_ports	Array	Array of http ports.

Attribute	Type	Description
https_ports	Array	Array of https ports.

Infoblox Config

enable_network_record	bool	Default value is True. If false *network* type records are disabled.
enable_host_record	bool	Default value is True. If false *host* type records are disabled.
enable_a_record	bool	Default value is True. If false *a* type records are disabled.
enable_aaaa_record	bool	Default value is True. If false *aaaa* type records are disabled.

** Read-only status fields in the Orchestrator object **

Attribute	Type	Description
authentication_failure	bool	Status of the connection to the Secure Workload Orchestrator - *true* indicates a successful connection to the orchestrator. If this field is *false*, the *authentication_failure_error* field will provide a detailed error message explaining the reason for the failure
authentication_failure_error	string	Detailed error message to help debug connectivity or credential failures with orchestrators
scope_id	string	Tenant Root scope id where the inventory will be published and visible

Get Orchestrators

This endpoint returns a list of orchestrators that are known to Secure Workload appliance. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}
```

Parameters: None

Returns a list of orchestrator objects for the provided root scope. The *scope* MUST be a root scope id.

Create Orchestrators

This endpoint is used to create orchestrators.

```
POST /openapi/v1/orchestrator/{scope}
```

Sample python code for vCenter orchestrators

```
req_payload = {
    "name": "VCenter Orchestrator"
    "type": "vcenter",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 443}],
    "username":"admin",
    "password":"admin"
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for DNS orchestrators

```
req_payload = {
    "name": "DNS Server"
    "type": "dns",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "dns_zones": [ "lab.corp.com.", "dev.corp.com." ]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for Kubernetes orchestrators

```
req_payload = {
    "name": "k8s"
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for Kubernetes orchestrators with Ingress Controller

See information about the Kubernetes/OpenShift external orchestrator for creating authentication details.

```
req_payload = {
    "name": "k8s",
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
    "ingress_controllers": [
        {
            "pod_selector": {
                "namespace": "ingress-nginx",
                "labels": [{ "key": "app", "value": "nginx-ingress"}],
            }
        }
    ]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for Kubernetes orchestrators with Multiple Ingress Controllers

See information about the Kubernetes/OpenShift external orchestrator for creating authentication details.

```
  req_payload = {
      "name": "k8s",
      "type": "kubernetes",
      "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
      "certificate": "",
      "key": "",
      "ca_certificate": "",
      "ingress_controllers": [
          {
              "pod_selector": {
                  "namespace": "ingress-nginx",
                  "labels": [{ "key": "app", "value": "nginx-ingress"}],
              },
              "controller_config": {
                  "ingress_class": "nginx-class",
              }
          },
          {
              "pod_selector": {
                  "namespace": "ingress-haproxy",
                  "labels": [{ "key": "app", "value": "haproxy-ingress"}],
              },
              "controller_config": {
                  "ingress_class": "haproxy-class",
                  "http_ports": [8080],
                  "https_ports": [8443],
                  "namespace": "haproxy-watching-namespace"
              }
          }
      ],
  }
  resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))

  ** Type AWS and EKS are no longer supported in external orchestrators. They have been
ported to
     connectors.
```

Get Specific Orchestrator

This endpoint returns an orchestrator instance.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Returns the orchestrator object that is associated with the specified ID.

Update an Orchestrator

This endpoint updates an orchestrator.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Parameters:

Same as POST parameters

Delete Specific Orchestrator

This endpoint deletes the specified orchestrator.

```
DELETE /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Orchestrator Golden Rules

This set of APIs can be used to manage Golden Rules for external Kubernetes Orchestrators. Golden Rules are necessary to ensure Kubernetes control plane connectivity in allow list enforcement mode. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator type for Golden Rules is 'kubernetes' only. Requests to this endpoint for non- Kubernetes orchestrators will fail.

Orchestrator Golden Rules Object

The orchestrator object attributes are described below:

Attribute	Type	Description
kubelet_port	integer	Kubelet node-local API port
services	Array	Array of Kubernetes Services objects

Get Orchestrator Golden Rules

This endpoint returns the golden rules that are associated with an orchestrator. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}/{id}/gr
```

Parameters: None

Returns a single Golden Rules object

Create or Update Golden Rules

This endpoint is used to create or update golden rules for an existing orchestrator.

```
POST /openapi/v1/orchestrator/{scope}/{id}/gr
```

Parameters:

Attribute	Type	Description
kubelet_port	integer	Kubelet node-local API port
services	Array	Array of Kubernetes Services objects

Sample python code

```
  req_payload = {
              "kubelet_port":10255,
              "services": [
                        {
    "description": "kube-dns",
    "addresses":   [ "10.0.1.1:53/TCP", "10.0.1.1:53/UDP" ],
    "consumed_by":  [ "NODES", "PODS"],
   }
             ]
  }
  resp = restclient.post('/orchestrator/{scope_id}/{orchestrator_id}/gr',
json_body=json.dumps(req_payload))
```

FMC Orchestrator Domains

This set of APIs can be used to manage domains for external FMC Orchestrators. FMC Domains are required to enable enforcement on a given FMC Domain. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator type for FMC Domains is ‘fmc’ only. Requests to this endpoint for non-FMC orchestrators will fail.

Orchestrator FMC Domains Object

The orchestrator object attributes are described below:

Attribute	Type	Description
fmc_domains	Array	Array of FMC Domain objects

FMC Domain object attributes are described below:

Attribute	Type	Description
name	string	Name of FMC Domain
enforcement_enabled	boolean	This flag is set to false by default. If true, the external orchestrator will deploy policies to the domain matching with ‘name’ when policy enforcement is performed for the workspace.

URL attributes are described below:

Attribute	Type	Description
scope	string	Tenant Root scope name or id where the inventory will be published and visible

Attribute	Type	Description
orchestrator_id	string	Orchestrator ID of FMC orchestrator

Get FMC Domains

This endpoint returns the FMC domains that are configured on the FMC that is associated with an FMC orchestrator. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}/fmcdomains
```

Parameters: None

Returns a JSON object with a list of FMC Domain object attributes.

Update FMC Domain Configuration for FMC External Orchestrator

This endpoint updates the FMC domain attributes for an existing FMC external orchestrator.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}/fmcdomains
```

Parameters:

Attribute	Type	Description
fmc_domains	Array	Array of FMC Domain objects

FMC Domain object attributes are described below:

Attribute	Type	Description
name	string	Name of FMC Domain
enforcement_enabled	boolean	This flag is set to false by default. If true, the external orchestrator deploys policies to the domain matching with ‘name’ when policy enforcement is performed for the workspace.

URL attributes are described below:

Attribute	Type	Description
scope	string	Tenant Root scope name or id where the inventory will be published and visible
orchestrator_id	string	Orchestrator ID of FMC orchestrator

Sample python code

```
  req_payload = {
        "fmc_domains":    [
        {
            "enforcement_enabled": False,
            "name": "Global/Eng"
        },
        {
            "enforcement_enabled": True,
            "name": "Global/Prod"
        }
    ]
  }
  resp = restclient.put('/orchestrator/{scope}/{orchestrator_id}/fmcdomains',
json_body=json.dumps(req_payload))
```

RBAC (Role-Based Access Control) Considerations

Access to orchestrators under a root scope requires that the API Key that is used for the request has the requisite privileges. All orchestrator API calls are scoped and always require the root scope id as part of the URL. Orchestrators always reside at the root scope level and cannot be created under subscopes. Orchestrators created (and inventory that is learned by these orchestrators) under a specific tenant root scope are invisible to other tenants.

In case of F5 load balancers that may have multiple route domains (vrfs) configured, the F5 Route Domain filtering logic scans all entities on the F5 across all partitions but discard entities (services, snat pools, pools, and backends) that do not evaluate to the route domain specified in the F5 orchestrator *route_domain* field.

High Availability and Failover Considerations

The hosts_list parameter allows configuration of multiple server addresses for an orchestrator. Secure Workload server selection logic in the case of multiple server addresses varies for each orchestrator type.

For *vCenter, Kubernetes, DNS, F5, Netscaler, Infoblox*, the selection is on a first healthy endpoint basis. Connections are not persistent (except for *kubernetes*) and thus, every poll period, Secure Connector Orchestrator Manager will scan the hosts and poll the first healthy endpoint encountered in the hosts_list. For *kubernetes*, a persistent event channel is maintained and upon connection failure, a scan of all hosts and subsequent full poll will be performed using the next healthy endpoint.

Kubernetes RBAC Resource Considerations

The Kubernetes client attempts to GET/LIST/WATCH the following resources.

The provided Kubernetes authentication credentials should have a minimum set of privileges to the following resources:

Resources	Verbs
daemonsets	[get list watch]
deployments	[get list watch]

Resources	Verbs
endpoints	[get list watch]
namespaces	[get list watch]
nodes	[get list watch]
pods	[get list watch]
replicasets	[get list watch]
replicationcontrollers	[get list watch]
services	[get list watch]
statefulsets	[get list watch]
daemonsets.apps	[get list watch]
deployments.apps	[get list watch]
endpoints.apps	[get list watch]
namespaces.apps	[get list watch]
nodes.apps	[get list watch]
pods.apps	[get list watch]
replicasets.apps	[get list watch]
replicationcontrollers.apps	[get list watch]
services.apps	[get list watch]
statefulsets.apps	[get list watch]
daemonsets.extensions	[get list watch]
deployments.extensions	[get list watch]
endpoints.extensions	[get list watch]
namespaces.extensions	[get list watch]
nodes.extensions	[get list watch]
pods.extensions	[get list watch]
replicasets.extensions	[get list watch]
replicationcontrollers.extensions	[get list watch]
services.extensions	[get list watch]
statefulsets.extensions	[get list watch]

Service Health

This API can be used to get the health of all services that are used in the Secure Workload cluster along with their dependencies.

Note This API is only available to site admin users.

Get Service Health

This endpoint returns a JSON object with service health information.

```
GET /openapi/v1/service_status
```

Parameters: None

Response object: JSON object with service health information

Sample Python code

```
resp = restclient.get('/service_status')
```

Secure Connector

OpenAPI exposes the endpoints to manage the functions of the Secure Connector. These endpoints require the `external_integration` capability to be associated with the API key.

Note The Secure Connector APIs cannot be used at site level. They can only be used at the root scope level.

Get Status

This endpoint returns the current status of the Secure Connector Tunnel for the specified root scope.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/status
```

```
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/status
```

READ permission to the specified root scope is required.

The returned status is a json object with the following schema:

Key	Type	Value
active	boolean	A Secure Connector tunnel is currently active
peer	string	<ip>:<port> of the Secure Connector client end of the tunnel

Key	Type	Value
start_time	int	Timestamp at which the tunnel was started (epoch time in seconds)
last_heartbeat	int	Timestamp of last heartbeat from the client (epoch time in seconds)

Get Token

This endpoint returns a new single-use limited-time token to be used for bootstrapping a Secure Connector client for the specified root scope.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/token
```

```
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/token
```

OWNER permission to the specified root scope is required.

The returned token is a string which contains a cryptographically signed token that is valid for one hour. A valid token can be used only once to bootstrap a Secure Connector client.

Rotate Certificates

This endpoint forces the creation of a new certificate for the specified root scope. The new certificate will be used by the Secure Connector server and will be used to sign the certificate signing requests from clients for this root scope.

```
POST /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/rotate_certs?invalidate_old=
‹→{true|false}
```

```
POST /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/rotate_certs?invalidate_old= ‹→{true|false}
```

OWNER permission to the specified root scope is required.

Once this endpoint is called, communication between the client and server for this root scope will immediately transition to using the new certificate.

If *invalidate_old* is set to false, any existing clients will automatically create a new public/private key pair and use their existing certificates to sign a new certificate for the new public key.

If *invalidate_old* is set to true, the existing certificate will be immediately invalidated. Any existing clients will not be able to connect to the server and will have to be bootstrapped once again using a new token. See Secure Connector Deployment for more information.

Kubernetes Vulnerability Scanning

Get Kubernetes Registries used for Pod Vulnerability Scanning

This endpoint returns a list of all Kubernetes registries displayed in the Secure Workload cluster for a given VRF.

```
GET /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/registry
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
root_scope_name_or_id	string	Root scope name or ID

Response object: Returns an array of registry objects with the following attributes

Attribute	Type	Description
id	string	ID of the registry
clusters	array	Array of Kubernetes cluster objects using the registry
connection_status	string	Defines registry connection status
credential_status	string	Status stating if the credentials are provided
last_scanned	Int64	Last scanner at time in epoch
url	string	URL of the registry

Kubernetes cluster object

Attribute	Type	Description
id	string	ID of the Kubernetes cluster
connector_id	string	ID of the connector used to onboard the Kubernetes cluster
connector_type	string	Type of connector used to onboard the Kubernetes cluster
name	string	Name of the Kubernetes cluster

Sample python code

```
root_app_scope_name = 'Tetration'
restclient.get('/kubernetes/%s/vulnerability_scanning/registry' % root_app_scope_name)
```

Add Credentials to Kubernetes Registry

This endpoint allows you to add credentials for the Kubernetes registry. The accepted credentials are based on the registry type.

For example:

Registry type: AWS; Accepted credential type: aws_auth Credentials object

Registry type: OTHER; Accepted credential type: basic_auth Credentials object

```
PUT /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/registry/{registry_id}
```

Parameters: The JSON query body contains the following keys:

Name	Type	Description
root_scope_name_or_id	string	Root scope name or ID
registry_id	string	Kubernetes registry ID
registry_credential	object	Credential object

Credential object

Attribute	Type	Description
basic_auth	object	Basic authentication credential object
aws_auth	object	AWS authentication credential object
azure_auth	object	Azure authentication credential object
gcp_auth	object	GCP authentication credential object

basic_auth object

Attribute	Type	Description
username	string	Username
password	string	Password

aws_auth object

Attribute	Type	Description
aws_access_key_id	string	AWS credentials access key
aws_secret_access_key	string	AWS credentials access secret

azure_auth object:

Attribute	Type	Description
azure_tenant_id	string	Azure tenant ID
azure_client_id	string	Azure client ID
azure_client_secret	string	Azure client secret

gcp_auth object:

Attribute	Type	Description
gcp_service_account	object	GCP service account

Sample python code

```
root_app_scope_name = 'Tetration'
registry_id = '64cdc7a7362f57192dcc1625'
pay_load = {
        "registry_credential": {
            "basic_auth": {
                "username": "username",
                "password": "password",
            }
        }
    }
restclient.put('/kubernetes/%s/vulnerability_scanning/registry/%s' % root_app_scope_name,
registry_id, json_body=json.dumps(pay_load))
```

Get Kubernetes Pod Scanners

This endpoint returns a list of all Kubernetes pod scanners displayed in the Secure Workload cluster for a given VRF.

```
GET /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/scanner
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
root_scope_name_or_id	string	Root scope name or ID

Response object: Returns an array of registry objects with the following attributes

Attribute	Type	Description
id	string	ID of the scanner
kubernetes_cluster	object	Add Credentials to Kubernetes Registry
health_status	string	Defines scanner health state
health_object	string	Health status object
scanner_action	string	Notifies if the scanner is ENABLED or DISABLED
name	string	Name of the scanner

Health object

Attribute	Type	Description
last_checkin	string	Last reported at time in epoch

Attribute	Type	Description
scanner_sensor_name	string	Kubernetes node name on which the scanner is running
scanner_sensor_uuid	string	ID of the agent running on the Kubernetes node
status	string	Notifies if health is reported by the scanner

Sample python code

```
root_app_scope_name = 'Tetration'
restclient.get('/kubernetes/%s/vulnerability_scanning/scanner % root_app_scope_name)
```

Edit Scanner Filter Query and Action

This endpoint allows you to edit Kubernetes scanner filter query and action.

```
PUT /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/scanner/{scanner_id}
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
rootAppScopeName	string	Root scope name or ID
scanner_id	string	Kubernetes scanner ID
scanner_action	string	To enable or disable the scanner. Expected values are ENABLED or DISABLED
filter_query	object	Validate an inventory filter query to filter pods for vulnerability scanning.

Sample python code

```
root_app_scope_name = 'Tetration'
scanner_id = '64cdc7a7362f57192dcc1625'
pay_load = {
        "scanner_action": "ENABLED"
        "filter_query": {
            "type": "contains",
            "field": "user_orchestrator_system/pod_name",
            "value": "pod"
        }
}
restclient.put('/kubernetes/%s/vulnerability_scanning/scanner/%s' % root_app_scope_name,
scanner_id, json_body=json.dumps(pay_load))
```

Policy Enforcement Status for External Orchestrators

This set of APIs is used to provide policy enforcement status for load balancer external orchestrators such as *F5 BIG-IP* or *Citrix Netscaler*.

Note To use these APIs, you must have access to the scope attached to the VRF.

Get Policy Enforcement Status for All External Orchestrators

This endpoint returns policy enforcement status for all external orchestrators belonging to the given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/tnp_policy_status/{vrfID}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
vrfID	integer	VRF ID for the root scope.

Response object: Returns a list of network policies with the Status as `ENFORCED` or `FAILED` or `IGNORED`.

Sample python code

```
vrf_id = 676767
restclient.get('/tnp_policy_status/%d' % vrf_id)
```

Get Policy Enforcement Status for an External Orchestrator

This endpoint returns policy enforcement status for an external orchestrator belonging to the given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/tnp_policy_status/{vrfID}/{orchestratorID}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
vrfID	integer	VRF ID for the root scope.
orchestratorID	string	External orchestrator ID.

Response object: Returns a list of network policies with the Status as `ENFORCED` or `FAILED` or `IGNORED`.

Sample python code

```
vrf_id = 676767
orchestrator_id = '5ee3c991497d4f3b00f1ee07'
restclient.get('/tnp_policy_status/%d/%s' % (vrf_id, orchestrator_id))
```

Download Certificates for Managed Data Taps and Datasinks

This set of APIs is used to download the certificates for the Managed Data Taps and Datasinks.

Note To use these APIs, you must have access to the scope attached to the VRF.

Get List of Managed Data Taps for a Given VRF ID.

This endpoint returns a list of Managed Data Taps in a given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/mdt/{vrfID}
```

Parameters: None

Returns a list of Managed Data Taps with attributes like Managed Data Tap ID.

Download Managed Data Tap Certificates for a Given MDT ID

This endpoint is used to download the certificates for a given Managed Data Tap ID. The MDT ID can be obtained by using **/openapi/v1/mdt/{vrfID}** endpoint as explained in the above documentation. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/mdt/{vrfID}/{mdtID}/certs
```

Parameters:

Name	Type	Description
format	string	The keystore and trust store format. Values: `jks`(default value) or `cert`

Returns a tar.gz file which contains the following files:-

For `jks` format: **truststore.jks, topic.txt, passphrase.txt, keystone.jks, kafkaBrokerIps.txt, consumer_name.txt, consumer_group_id.txt.**.

For `jks` format: **KafkaConsumerCA.cert, KafkaConsumerPrivateKey.key, kafkaCA.cert, kafkaBrokerIps.txt, topic.txt**

KafkaConsumerCA.cert is the Public certificate file and **KafkaConsumerPrivateKey.key** file has the private key. **kafkaCA.cert** has the CA certificate and **kafkaBrokerIps.txt** has the list of the Kafka brokers IP Addresses and Ports. topic.txt file has the name of the topic which should be used to fetch data from MDT. **truststore.jks** and **keystone.jks** are Java keystore files.

Get List of DataSinks for a Given VRF ID

This endpoint returns a list of DataSinks in a given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/datasinks/{vrfID}
```

Parameters: None

Returns a list of DataSinks with attributes like DataSink ID.

Download DataSink Certificates for a Given DataSink ID

This endpoint is used to download the certificates for a given DataSink ID. The DataSink ID can be obtained by using **/openapi/v1/datasinks/{vrfID}** endpoint as explained in the above documentation. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/datasinks/{vrfID}/{dsID}/certs
```

Parameters: None

Returns a tar.gz file which contains the following files:- **userCA.cert, userPrivateKey.key, intermediateCA.cert, kafkaCA.cert, kafkaBrokerIps.txt, topic.txt**.

userCA.cert is the Public certificate file and **KafkaConsumerPrivateKey.key** file has the private key. **intermediateCA. cert** and **kafkaCA.cert** has the CA certificate for intermediate and root CA respectively. **kafkaBrokerIps.txt** has the list of the Kafka brokers IP Addresses and Ports. **topic.txt** file has the name of the topic which should be used to fetch data from datasink.

Change Logs

This API provides read access to change log items. This API requires the `user_role_scope_management` capability associated with the API key.

Note This API is only available to site admins and owners of root scopes.

Change Log Object

The descriptions of the change log object attributes are as follows:

Attribute	Type	Description
id	string	Unique identifier for the change log item.
association_chain	array of objects	List of names and ids associated with this change.
scope	string	Scope of change (not the same as a Secure Workload scope).
action	string	Change action.
details	string	Further action details, when available.

Attribute	Type	Description
created_at	integer	Unix timestamp of when change log item was created.
modifier	object	User responsible for change.
modified	object	Modified fields and values.
original	object	Fields and values before modification.
version	integer	Version identifier.

Search

This endpoint returns the list of change log items matching the specified criteria.

```
GET /openapi/v1/change_logs
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_app_scope_id	string	(optional) Required for root scope owners. Filter results by root scope.
association_name	string	(optional) Required for root scope owners. The item type to return. For example: “H4Users”
history_action	string	(optional) Change action. For example: “update”
details	string	(optional) Action details. For example: “soft-delete”
before_epoch	integer	(optional) Include results created before this unix timestamp.
after_epoch	integer	(optional) Include results created after this unix timestamp.
offset	integer	(optional) Number of results to skip.
limit	integer	(optional) Limit number of results, default 1000.

Response object: Returns a list of change log objects.

Response

The response is a JSON object in the body with the following properties.

Name	Type	Description
total_count	integer	Total number of items matching before applying offset or limit.
items	array of objects	List of results.

Sample python code

Fetch last 100 scope object changes within a given root scope within the last day.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24*60*60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'after_epoch': one_day_ago,
                                              'limit': 100})
```

Fetch the second thousand scope object changes.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'offset': 1000})
```

Further refine these results to only show new scope creations.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24 * 60 * 60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'history_action': 'create',
                                              'after_epoch': one_day_ago,
                                              'limit': 100})
```

A site admin could use limit and offset to iteratively fetch all changes across all scopes.

```
resp = restclient.get('/change_logs', params={'offset': 100, 'limit': 100})
```

Non-Routable Endpoints

The following APIs are used to manage nonroutable endpoints, to mark an IP or subnet as nonroutable or get a list of nonroutable endpoints that are marked by a user or to unmark an IP or subnet as nonroutable endpoint. The `user_data_upload` capability that is associated with the API key is required.

Non-Routable Endpoint Object

The descriptions of the Non-Routable Endpoint Object attributes are as follows:

Attribute	Type	Description
id	string	Unique identifier for the nonroutable endpoint.
name	string	User specified name of the nonroutable endpoint.
subnet	string	IPv4/IPv6 subnet.

Attribute	Type	Description
vrf_id	long	ID of the VRF to which the nonroutable endpoint belongs to.
address_type	string	IPV4/IPV6 based on subnet address type
host_uuid	string	Unique ID of the agent
description.	string	User specified description of the nonroutable endpoint.

GET Non-Routable Endpoints

This endpoint returns a list of nonroutable endpoints in the given tenant.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters: None

Create a Non-Routable Endpoint

This endpoint is used to create a nonroutable endpoint.

```
POST /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters:

Attribute	Type	Description
name	string	User specified name of the nonroutable endpoint.
subnet	string	IPv4 or IPv6 subnet.
address_type(optional)	string	IPv4or IPv6 based on subnet address type
host_uuid(optional)	string	Unique ID of the agent
description(optional)	string	User specified description of the nonroutable endpoint.

*if optional fields are not specified, null values are populated.

Sample python code

```
req_payload = {
  "name": "nre-1",
  "subnet": "1.1.1.1/30",
  "address_type": IPV4,
  "description": "sample parameters test"
}
```

```
  resp = restclient.post('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

GET Specific Non-Routable Endpoints with Name

This endpoint returns a nonroutable endpoint for the specified name.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

Parameters: None

GET Specific Non-Routable Endpoints with ID

This endpoint returns a nonroutable endpoint for the specified ID.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

Parameters: None

Update Specific Non-Routable Endpoint Name

This endpoint is used to update a nonroutable endpoint. It uses either an ID or a name of the existing nonroutable endpoint to update its name.

```
PUT /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters:

Attribute	Type	Description
id	string	Unique identifier for the nonroutable endpoint.
name	string	User specified name of the nonroutable endpoint.
new_name	string	New name to update

Sample python code

```
  req_payload = {
    "name": "nre-1",
    "new_name": "nre-updated",
  }
  resp = restclient.put('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

```
  req_payload = {
    "id": "5f706964a5b5f16ed4b0aacb",
    "new_name": "nre-updated",
  }
  resp = restclient.put('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

Delete Specific Non-Routable Endpoint with Name

This endpoint deletes the specific nonroutable endpoint.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

Delete Specific Non-Routable Endpoint with ID

This endpoint deletes the specific nonroutable endpoint.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

Config and Command Schemas for External Appliances and Connectors

Config Groups APIs

The Config Groups APIs provides config schemas for the Appliances and Connectors APIs. These APIs require the `sensor_management` or `external_integration` capability that is associated with the API key.

API to Get the Schema of Config

This endpoint returns a static config schema for selected type/groups of configs.

```
GET /openapi/v1/config_groups/schema/<type>
```

where <type> is the Appliance config type.

Parameters: The request URL contains the following parameters

Name	Type	Description
type	string	Specify the config type from "VM1" "VM3" "NTP" "LOG" "LDAP" "NETFLOW" "IPFIX" "NETSCALER" "F5" "AWS" "ENDPOINT" "SLACK_NOTIFIER" "GCP_CONNECTOR" "PAGERDUTY_NOTIFIER" "SYSLOG_NOTIFIER" "KINESIS_NOTIFIER" "EMAIL_NOTIFIER" "ISE" "MERAKI" "SLACK_NOTIFIER_OVERRIDE" "PAGERDUTY_NOTIFIER_OVERRIDE" "SYSLOG_NOTIFER_OVERRIDE" "KINESIS_NOTIFER_OVERRIDE" "AZURE_CONNECTOR" "EMAIL_NOTIFIER_OVERRIDE" "SYSLOG_SEVERITY_MAPPING" "SERVICENOW" "SYNC_INTERVAL" "ALERT" "VM3_ERSPAN" "AWS_CONNECTOR" "VM0"

Response object: Returns the config schema for selected config type.

Sample response

```
resp = restclient.get('/config_groups/schema/LOG')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "type": "LOG",
    "name": "Log",
    "mode": "TEST",
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20 }
    },
    "fill_ins": [
      {
        "field": "log-level",
        "label": "Logging Level",
        "placeholder": "info",
        "type": "user_fill_in",
        "input_type": "dropdown",
        "possible_values": [
          "trace",
          "debug",
          "info",
          "warn",
          "error"
        ]
    },
      {
        "field": "max-log-size",
        "label": "Max Log File Size (in MB)",
        "placeholder": 10,
        "type": "user_fill_in",
        "input_type": "number",
        "min": 1,
        "max": 10240
      },
      {"field": "max-log-age",
        "label": "Log Rotation (in days)",
        "placeholder": 30,
        "type": "user_fill_in",
        "input_type": "number",
        "min": 1,
        "max": 365
      },
      {
        "field": "max-log-backups",
        "label": "Log Rotation (in instances)",
        "placeholder": 20,
        "type": "user_fill_in",
        "input_type": "number",
        "min": 1,
        "max": 100
}
    ]
  }
```

API to Get the Schema of Troubleshooting Commands

This endpoint returns a static troubleshooting command config schema for selected type of troubleshooting command.

```
GET /openapi/v1/config_groups/command_schema/<type>
```

In request payload, <type> is the troubleshooting command config type.

Parameters:

The request URL contains the following parameters

Name	Type	Description
type	string	Specify the command type from "SHOW LOG" "SHOW_SERVICE_LOG" "SHOW_RUNNING_CONF" "SHOW_SERVICE_RUNNING_CONF" "SHOW_SYS_COMMANDS" "SHOW_DOCKER_COMMANDS" "SHOW_DOCKER_INSTANCE_COMMANDS" "OPER_DOCKER_INSTANCE_COMMANDS" "SHOW_SUPERVISOR_COMMANDS" "SHOW_SUPERVISOR_SERVICE_COMMANDS" "OPER_SUPERVISOR_SERVICE_COMMANDS" "NETWORK_CONNECTIVITY_COMMANDS" "LIST_FILES" "LIST_SERVICE_FILES" "PACKET_CAPTURE" "SHOW_DATA_EXPORT_LGO" "SHOW_DATAEXPORT_RUNNING_CONF" "SHOW_DATA_EXPORT_SYS_COMMANDS" "UPDATE_LISTENING_PORT" "UPDATE_TAN_LOG_CONF" "SNAPSHOT_APPLIANCE" "SNAPSHOT_CONNECTOR" "SHOW_AWS_DOWNLOADER_LOG" "CONTROLLER_PROFILING" "SERVICE_PROFILING" "RESTART_CONNECTOR_CONTAINER" "RESTART_CONNECTOR_SERVICE" "CONNECTOR_ALERT_INTERVAL_APPLIANCE" "CONNECTOR_ALERT_INTERVAL_CONNECTOR" "EXEC_SCRIPT" "SHOW_SEGMENTATION_POLICIES"

Response object: Returns the config schema for selected config type.

Sample response

```
resp = restclient.get('/config_groups/command_schema/SHOW_LOG')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "name": "Show logs",
    "desc": "Show the contents of a log file",
   "long_desc": "Show the contents of a log file and optionally grep the file for a specified
 pattern. The output is tailed for the last 5000 lines.",
    "valid_appliances": [
      "TETRATION_DATA_INGEST",
      "TETRATION_EDGE",
      "TETRATION_EXPORT"
    ],
    "valid_connectors": [
      "netflow",
      "netscaler",
      "f5",
      "aws",
      "anyconnect",
      "slack",
      "kinesis",
      "syslog",
      "email",
      "pagerduty",
      "ise",
      "asa",
      "meraki",
      "servicenow",
      "wad"
    ],
    "arg_fillins": [
      {
        "field": "pattern",
        "label": "Grep Pattern",
        "input_type": "text",
        "optional": true
      }
    ],
    "output_type": "FILE",
    "output_ext": "LOG"
  }
```


External Appliances

External Appliances APIs

The External Appliances APIs are associated with managing SecureWorkload external Appliances. These set of APIs require the `sensor_management` or `external_integration` capability associated with the API key.

API to Get List of Appliances

This endpoint returns the list of Appliances.

```
GET /openapi/v1/ext_appliances?root_scope_id=<root_scope_id>&type=<type>
```

where <root_scope_id> is the root_scope_id that can be obtained from the Get scopes API, <type> is a string to decide Appliance type.

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Specify the root scope

Name	Type	Description
type	string	Specify the Appliance type, the value can be either “TETRATION_EDGE”, “TETRATION_DATA_INGEST”, “TETRATION_EXPORT”, “TETRATION_ERSPAN” or “TETRATION_INTERNAL”

Response object: Returns the list of Appliance.

Sample response

```
resp =
restclient.get('/ext_appliances?root_scope_id=63bf8d2f497d4f7287dbd335&type=TETRATION_INTERNAL')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
[
    {
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "type": "tetration_internal",
      "status": {
        "state": "active",
        "controller_state": "up",
        "message": "",
        "display_state": "active"
      },
      "auto_upgrade": true,
      "created_at": 1673498141,
      "updated_at": 1673498141,
      "registered_at": 1673498141,
      "last_checkin_at": 0,
      "last_rpm_sent_at": 0,
      "upgrade_attempts": 0,
      "delete_attempts": 0,
      "last_delete_msg_sent_at": 0,
      "taas": false,
      "deleted": false,
      "deleted_at": 0,
      "connector_limit": 5000,
      "available_slots": 5000,
      "internal": true,
      "id": "63bf8e1d6419d06bef39bc85",
      "ha_peer_appliance_id": "",
      "display_type": "Tetration Internal"
    }
  ]
```

API to Create an Appliance

This endpoint creates an Appliance.

```
POST /openapi/v1/ext_appliances
```

In request payload, to obtain <config> schema, select one of the “valid_config” from API to Get Appliance Schema, on page 184 response, apply the “valid_config” to API to Get the Schema of Config, on page 171.

Parameters: The request URL contains the following parameters

Name	Type	Description
name	string	Specify the name
root_scope_id	string	Specify the root scope
type	string	Specify the Appliance type, the value can be either “TETRATION_EDGE”, “TETRATION_DATA_INGEST”, “TETRATION_EXPORT”, “TETRATION_ERSPAN” or “TETRATION_INTERNAL”
config	set	Provide the filled config schema in JSON format
taas	boolean	Indicate if the Appliance is for TAAS env
version	string	Specify the version

Response object: Returns the created Appliance.

Sample response

```
req_payload = {
    "name":"Data Ingest Appliance",
    "type":"tetration_data_ingest",
    "root_scope_id":"63c41ff2497d4f5f5be73662",
    "config":{
      "VM3":{
        "secured":{},
        "unsecured":{
          "cidr":[
            "172.26.231.141/23",
            "172.26.231.142/23",
            "172.26.231.143/23"
          ],
          "gateway":[
            "172.26.231.140",
            "172.26.231.140",
            "172.26.231.140"
          ],
          "cidr_v6":[],
          "gateway_v6":[],
          "dns":[
            "testserver"
          ],
          "search_domains":[],
          "hostname":"",
          "use_proxy_for_tetration":false,
          "https_proxy":"",
          "no_proxy":[],
          "docker_subnet":""
        }
      }
    }
```

```
}
resp = restclient.post('/ext_appliances', json_body=json.dumps(req_payload))
if resp.status_code == 200:
  parsed_resp = json.loads(resp.content)
  print json.dumps(parsed_resp)
```

Sample response


```
{
   "root_scope_id":"63c41ff2497d4f5f5be73662",
   "type":"tetration_data_ingest",
   "status":{
     "state":"pending_dio",
     "controller_state":"down",
     "message":"Setting up appliance",
     "display_state":"preparing"
   },
   "auto_upgrade":true,
   "created_at":1674183549,
   "updated_at":1674183549,
   "registered_at":0,
   "last_checkin_at":0,
   "last_rpm_sent_at":0,
   "upgrade_attempts":0,
   "delete_attempts":0,
   "last_delete_msg_sent_at":0,
   "name":"Data Ingest Appliance",
   "taas":false,
   "deleted":false,
   "deleted_at":0,
   "connector_limit":3,
   "available_slots":0,
   "internal":false,
   "id":"63ca037dbca44e263daeb5d0",
   "ha_peer_appliance_id":"",
   "display_type":"Tetration Data Ingest"
}
```


API to Delete an Appliance

This endpoint deletes the given Appliance.

```
DELETE /openapi/v1/ext_appliances/<id>
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID

Response object: Returns the status of the deleted Appliance.

Sample response

```
resp = restclient.delete('/ext_appliances/63be3b1ade36423c12bff6e1')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response


```
{
   "status": 200,
```

```
    "code": 1000,
    "message": "deleted"
  }
```

API to Get an Appliance by ID

This endpoint gets the Appliance with given ID.

```
GET /openapi/v1/ext_appliances/<id>
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID

Response object: Returns the Appliance with given ID.

Sample response

```
resp = restclient.get('/ext_appliances/63bf8e1d6419d06bef39bc85')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "type": "tetration_internal",
    "status": {
      "state": "active",
      "controller_state": "up",
      "message": "",
      "display_state": "active"
    },
    "auto_upgrade": true,
    "created_at": 1673498141,
    "updated_at": 1673498141,
    "registered_at": 1673498141,
    "last_checkin_at": 0,
    "last_rpm_sent_at": 0,
    "upgrade_attempts": 0,
    "delete_attempts": 0,
    "last_delete_msg_sent_at": 0,
    "taas": false,
    "deleted": false,
    "deleted_at": 0,
    "connector_limit": 5000,
    "available_slots": 5000,
    "internal": true,
    "id": "63bf8e1d6419d06bef39bc85",
    "ha_peer_appliance_id": "",
    "display_type": "Tetration Internal"
  }
```

API to Rename an Appliance

This endpoint renames the Appliance with given ID.

```
PUT /openapi/v1/ext_appliances/<id>
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
name	string	Specify the new name of the Appliance

Response object: Returns the Appliance with given ID and the new name.

Sample response

```
req_payload = {
    "name": "new_name",
  }
  resp = restclient.put('/ext_appliances/63bf8e1d6419d06bef39bc85',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "type": "tetration_internal",
    "status": {
      "state": "active",
      "controller_state": "up",
      "message": "",
      "display_state": "active"
    },
    "auto_upgrade": true,
    "created_at": 1673498141,
    "updated_at": 1673498141,
    "registered_at": 1673498141,
    "last_checkin_at": 0,
    "last_rpm_sent_at": 0,
    "upgrade_attempts": 0,
     "delete_attempts": 0,
    "last_delete_msg_sent_at": 0,
    "name": "new_name",
    "taas": false,
    "deleted": false,
    "deleted_at": 0,
    "connector_limit": 5000,
    "available_slots": 5000,
    "internal": true,
    "id": "63bf8e1d6419d06bef39bc85",
    "ha_peer_appliance_id": "",
    "display_type": "Tetration Internal"
  }
```

API to Get the Configs on Config Type

This endpoint gets the configs of the Appliance with given ID and config type.

```
GET /openapi/v1/ext_appliances/<id>/config?config_type=<config_type>
```

where <id> is the appliance_id that can be obtained from API to Get List of Appliances, on page 174, <config_type> is the "valid_config" that can be obtained from API to Get Appliance Schema, on page 184.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
config_type	string	Specify the config type. Refer to API to Get the Schema of Config, on page 171 for all possible values listed under Description

Response object: Returns the configs with given Appliance ID and config type

Sample response

```
resp =
restclient.get('/ext_appliances/63c1272039042a1c0ddd3e20/config?config_type=VM3_ERSPAN')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response


```
[
    {
      "mode": "TEST_AND_APPLY",
      "name": "VM3_ERSPAN",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "deleted": false,
      "type": "VM3_ERSPAN",
      "state": "COMMIT",
      "attempts": 0,
      "config": {
        "secured": {},
        "unsecured": {
          "cidr": [
            "172.26.231.141/23",
            "172.26.231.142/23",
            "172.26.231.143/23"
          ],
          "gateway": [
            "172.26.231.140",
            "172.26.231.140",
            "172.26.231.140"
          ],
          "cidr_v6": [],
          "gateway_v6": [],
          "dns": [
            "test"
          ],
          "search_domains": [],
          "hostname": "hjtest",
          "https_proxy": "",
          "no_proxy": []
        }
      },
      "push_to_dio_at": 0,
      "created_at": 1673602848,
```

```
      "updated_at": 1673602848,
      "discovery_completed_at": 0,
      "committed_at": 0,
      "delete_at": 0,
      "error_at": 0,
      "hidden": false,
      "discovery_phase": 0,
      "internal": false,
      "id": "63c1272039042a1c0ddd3e21"
    }
  ]
```

API to Add a New Config to External Appliance

This endpoint adds a new config to the Appliance with given ID

```
POST /openapi/v1/ext_appliances/<id>/config
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174. In request payload, <type> is the "valid_config" that can be obtained from API to Get Appliance Schema, on page 184. To obtain <config> schema, select one of the "valid_config" from API to Get Appliance Schema, on page 184 response, apply the "valid_config" to API to Get the Schema of Config, on page 171.

Parameters: The request URL contains the following parameters

Name	Type	Description
name	string	Specify the config name
type	string	Specify the config type. Refer to API to Get the Schema of Config, on page 171 for all possible values listed under Description
config	set	Provide the filled config schema in JSON format

Response object: Returns the updated config.

Sample response

```
req_payload = {
    "name": "new_config",
    "type": "VM3_ERSPAN",
    "config": {}
  }
  resp = restclient.post('/ext_appliances/63c1272039042a1c0ddd3e20/config',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
 {
    "prev_id": "63c1272039042a1c0ddd3e21",
    "mode": "TEST_AND_APPLY",
    "name": "new_config",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "appliance_id": "63c1272039042a1c0ddd3e20",
    "deleted": false,
```

```
    "type": "VM3_ERSPAN",
    "state": "COMMIT",
    "attempts": 0,
    "config": {
      "secured": {},
      "unsecured": null
    },
    "push_to_dio_at": 0,
    "created_at": 1673661042,
    "updated_at": 1673661042,
    "discovery_completed_at": 0,
    "committed_at": 0,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63c20a7239042a0991b871b7"
  }
```

API to Delete a Config

This endpoint deletes a config from given Appliance.

```
DELETE /openapi/v1/ext_appliances/<id>/config/<config_id>
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174, <config_id> is the id that can be obtained from API to Get the Configs on Config Type, on page 179.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
config_id	string	Specify the config ID

Response object: Returns the status of the config deleted for given Appliance.

Sample response

```
resp =
restclient.delete('/ext_appliances/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API to Get the Config

This endpoint gets the config with given ID

```
GET /openapi/v1/ext_appliances/<id>/config/<config_id>
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174, <config_id> is the id that can be obtained from API to Get the Configs on Config Type, on page 179.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
config_id	string	Specify the config ID

Response object: Returns the config with given ID.

Sample response

```
resp =
restclient.get('/ext_appliances/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
b
```

Sample response


```
{
    "mode": "TEST_AND_APPLY",
    "name": "VM3_ERSPAN",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "appliance_id": "63c1272039042a1c0ddd3e20",
    "deleted": false,
    "type": "VM3_ERSPAN",
    "state": "COMMIT",
    "attempts": 0,
    "config": {
      "secured": {},
      "unsecured": {
        "cidr": [
          "172.26.231.141/23",
          "172.26.231.142/23",
          "172.26.231.143/23"
        ],
        "gateway": [
          "172.26.231.140",
          "172.26.231.140",
          "172.26.231.140"
        ],
        "cidr_v6": [],
        "gateway_v6": [],
        "dns": [
          "test"
        ],
        "search_domains": [],
        "hostname": "hjtest",
        "https_proxy": "",
        "no_proxy": []
      }
    },
    "push_to_dio_at": 0,
    "created_at": 1673602848,
    "updated_at": 1673602848,
    "discovery_completed_at": 0,
    "committed_at": 0,
    "delete_at": 0,
```

```
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63c1272039042a1c0ddd3e21"
  }
```

API to Get Appliance Schema

This endpoint gets the Appliance schema with given type

```
GET /openapi/v1/ext_appliances/schema/<type>
```

where <type> is a string to decide Appliance type.

Parameters: The request URL contains the following parameters

Name	Type	Description
type	string	Specify the Appliance type, the value can be either "TETRATION_EDGE", "TETRATION_DATA_INGEST", "TETRATION_EXPORT", "TETRATION_ERSPAN" or "TETRATION_INTERNAL"

Response object: Returns the config schema.

Sample response

```
resp = restclient.get('/ext_appliances/schema/TETRATION_ERSPAN')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
 {
    "name": "Data Ingest for ERSPAN",
    "type": "tetration_erspan",
    "desc": "Data Ingest Appliance for ERSPAN",
    "long_desc": "Data Ingest appliance for ERSPAN is a software appliance that can export
 flow data from ERSPAN appliance.",
    "valid_config": [
      "VM3_ERSPAN"
    ],
    "deploy_config": [
      "VM3_ERSPAN"
    ],
    "connectors": [
      "ERSPAN"
    ],
    "limit_connectors_per_appliance": 0,
    "limit_per_rootscope": 8,
    "limit_per_rootscope_taas": 4,
    "limit_per_cluster": 150,
    "cco_url":
"https://software.cisco.com/download/home/286309796/type/286309874/release/3.7.1.26.devel"
  }
```

API to List Troubleshooting Commands Available for an Appliance

This endpoint returns the list of troubleshooting commands available for given Appliance.

```
GET /openapi/v1/ext_appliances/<id>/commands/available
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID

Response object: Returns the list of troubleshooting commands available for given Appliance.

Sample response

```
resp = restclient.get('/ext_appliances/63c6ef42bca44e2b5e729191/commands/available')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response


```
[
    {
      "type": "UPDATE_LISTENING_PORT",
      "name": "Update the listening port on a connector"
    },
    {
      "type": "SNAPSHOT_APPLIANCE",
      "name": "Collect Snapshot from Appliance"
    },
    {
      "type": "LIST_FILES",
      "name": "List a directory"
    },
    {
      "type": "CONTROLLER_PROFILING",
      "name": "Collect controller profile"
    },
    {
      "type": "SHOW_LOG",
      "name": "Show logs"
    },
    {
      "type": "SHOW_SUPERVISOR_COMMANDS",
      "name": "Execute supervisorctl command"
    },
    {
      "type": "PACKET_CAPTURE",
      "name": "Packet capture"
    },
    {
      "type": "NETWORK_CONNECTIVITY_COMMANDS",
      "name": "Test network connectivity"
    },
    {
      "type": "SHOW_DOCKER_COMMANDS",
      "name": "Execute docker command"
    },
    {
      "type": "CONNECTOR_ALERT_INTERVAL_APPLIANCE",
      "name": "Override connector alert interval for Appliance"
```

```
    },
    {
      "type": "SHOW_RUNNING_CONF",
      "name": "Show running configuration"
    },
    {
      "type": "SHOW_SYS_COMMANDS",
      "name": "Execute system command"
    }
  ]
```

API to List Troubleshooting Commands

This endpoint returns the list of troubleshooting commands that are enabled for given Appliance.

```
GET /openapi/v1/ext_appliances/<id>/commands
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID

Response object: Returns the list of troubleshooting commands that are enabled for given Appliance.

Sample response

```
resp = restclient.get('/ext_appliances/63be3b1ade36423c12bff6e1/commands')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
[
    {
      "appliance_id": "63be3b1ade36423c12bff6e1",
      "state": "pending",
      "level": "APPLIANCE",
      "command": "SHOW_LOG",
      "arg_string": "",
      "args": {},
      "tailed": false,
      "rc": 0,
      "push_to_dio_at": 0,
      "attempts": 0,
      "deleted": false,
      "deleted_at": 0,
      "created_at": 1673595392,
      "total_chunk": 0,
      "id": "63c10a0039042a6aee1b008c"
    }
  ]
```

API to Create a Troubleshooting Command

This endpoint creates a troubleshooting command available for given Appliance.

```
POST /openapi/v1/ext_appliances/<id>/commands
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174. In request payload, <command> is a troubleshooting command type that can be obtained from API to Get the

Schema of Troubleshooting Commands, on page 173 response "valid_appliances" field. <arguments> is a filled JSON object of the command schema, which can be obtained from API to Get the Schema of Troubleshooting Commands, on page 173.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
command	string	Specify the command type
arguments	set	Provide the filled command schema in JSON format

Response object: Returns the troubleshooting command created for given Appliance.

Sample response

```
req_payload = {
    "command": "SHOW_LOG",
    "arguments": {}
  }
  resp = restclient.post('/ext_appliances/63be3b1ade36423c12bff6e1/commands',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
   "appliance_id": "63be3b1ade36423c12bff6e1",
   "state": "pending",
   "level": "APPLIANCE",
   "command": "SHOW_LOG",
   "args": {},
   "tailed": false,
   "rc": 0,
   "push_to_dio_at": 0,
   "attempts": 0,
   "deleted": false,
   "deleted_at": 0,
   "created_at": 1673595392,
   "total_chunk": 0,
   "id": "63c10a0039042a6aee1b008c"
 }
```

-

API to Delete a Troubleshooting Command

This endpoint deletes a troubleshooting command enabled for given Appliance.

```
DELETE /openapi/v1/ext_appliances/<id>/commands/<command_id>
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174, <command_id> is the id that can be obtained from API to List Troubleshooting Commands, on page 186.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
command_id	string	Specify the command ID

Response object: Returns the status of the troubleshooting command deleted for given Appliance.

Sample response

```
resp =
restclient.delete('/ext_appliances/63be3b1ade36423c12bff6e1/commands/63c10a0039042a6aee1b008c')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
 {
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API to Return a Troubleshooting Command

This endpoint returns the selected troubleshooting command for a given Appliance.

```
GET /openapi/v1/ext_appliances/<id>/commands/<command_id>
```

where <id> is the appliance_id that is obtained from the API to Get List of Appliances, on page 174, <command_id> is the id that is obtained from API to List Troubleshooting Commands, on page 186.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
command_id	string	Specify the command ID

Response object: Returns the selected troubleshooting command for a given Appliance.

Sample response

```
resp =
restclient.get('/ext_appliances/63be3b1ade36423c12bff6e1/commands/63c10fd139042a1c0ddd3e1f')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "appliance_id": "63be3b1ade36423c12bff6e1",
    "state": "pending",
    "level": "APPLIANCE",
    "command": "SHOW_LOG",
    "arg_string": "",
    "args": {},
    "tailed": false,
```

```
    "rc": 0,
    "push_to_dio_at": 0,
    "attempts": 0,
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1673596881,
    "total_chunk": 0,
    "id": "63c10fd139042a1c0ddd3e1f"
}
```


API to Download the Output of the Appliance Command as a File

This endpoint downloads the output of the command as a file.

```
GET /openapi/v1/appliances/<id>/commands/{command_id}/download
```

where <id> is the appliance_id that can be obtained from the API to Get List of Appliances, on page 174, <command_id> is the id that can be obtained from API to List Troubleshooting Commands, on page 186. Not all commands have downloadable output, check the command schema provided by API to Get the Schema of Troubleshooting Commands, on page 173, where "output_type": "FILE" indicates it has downloadable content and "output_ext" tells the file type.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
command_id	string	Specify the command ID

Response object: Download the command output as a file.

Sample response

```
resp = restclient.download('downloadFile',
'/appliances/63c6ef42bca44e2b5e729191/commands/63cace941a49bd4c0e0cf45a/download')
```

Connectors

Connectors APIs

The Connectors APIs are associated with managing Secure Workload Connectors. These set of APIs require the `sensor_management` or `external_integration` capability associated with the API key.

API to Get All Types of Connectors

This endpoint gets all types of Connectors for given root scope.

```
GET /openapi/v1/connectors/cards?root_scope_id=<root_scope_id>
```

where <root_scope_id> is the root_scope_id that can be obtained from the Get scopes, on page 59 API.

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Specify the root scope

Response object: Returns all types of Connector with given root scope ID.

Sample response

```
resp = restclient.get('/connectors/cards?root_scope_id=63bf8d2f497d4f7287dbd335')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response


```
[{
    "type": "NETFLOW",
    "name": "NetFlow",
    "desc": "Collect telemetry from network devices",
    "long_desc": "Collect NetFlow V9 and/or IPFIX telemetry from network devices such as
routers and switches.",
    "group": "Flow Ingest",
    "index": 0,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "NETSCALER",
    "name": "NetScaler",
    "desc": "Collect telemetry from Citrix ADC",
   "long_desc": "Collect telemetry from Citrix ADC, stitch client and server side flows.",

    "group": "Flow Ingest",
    "index": 2,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Flow Stitching",
      "ADM"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "F5",
```

```
    "name": "F5",
    "desc": "Collect telemetry from F5 BIG-IP",
    "long_desc": "Collect telemetry from F5 BIG-IP, stitch client and server side flows,
enrich client inventory with user attributes.",
    "group": "Flow Ingest",
    "index": 1,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LDAP",
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "User Insights",
      "Flow Stitching",
      "ADM"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "ANYCONNECT",
    "name": "AnyConnect",
    "desc": "Collect telemetry from AnyConnect NVM",
    "long_desc": "Collect telemetry data from Cisco AnyConnect Network Visibility Module
(NVM) and enrich endpoint inventories with user attributes",
    "group": "Endpoints",
    "index": 0,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 50,
    "limit_per_cluster": 500,
    "config": [
      "ENDPOINT",
      "LDAP",
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Process Annotation",
      "User Insights",
      "Endpoint Insights",
      "Inventory Enrichment"
    ],
    "connector_count": 0,
    "group_order": 1
  },
  {
    "type": "ASA",
    "name": "Cisco Secure Firewall",
    "desc": "Collect telemetry from Cisco Secure Firewall",
```

```
    "long_desc": "Collect telemetry from Cisco Secure Firewall, stitch client and server
side flows. Recommended usage with ISE connector for flow visibility.",
    "group": "Flow Ingest",
    "index": 3,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Flow Stitching",
      "ADM"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "SLACK",
    "name": "Slack",
    "desc": "Send alerts to Slack",
    "long_desc": "Send Tetration Alerts to Slack.",
    "group": "Alerts",
    "index": 2,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "SLACK_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "KINESIS",
    "name": "Kinesis",
    "desc": "Send alerts to Kinesis",
    "long_desc": "Send Tetration Alerts to Kinesis.",
    "group": "Alerts",
    "index": 4,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "KINESIS_NOTIFIER",
      "ALERT"
```

```
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "SYSLOG",
    "name": "Syslog",
    "desc": "Send alerts to Syslog server",
    "long_desc": "Send Tetration Alerts to Syslog server.",
    "group": "Alerts",
    "index": 0,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "SYSLOG_NOTIFIER",
      "SYSLOG_SEVERITY_MAPPING",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "EMAIL",
    "name": "Email",
    "desc": "Send alerts to Email",
    "long_desc": "Send Tetration Alerts to Email.",
    "group": "Alerts",
    "index": 1,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "EMAIL_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "PAGERDUTY",
    "name": "Pager Duty",
```

```
    "desc": "Send alerts to Pager Duty",
    "long_desc": "Send Tetration Alerts to Pager Duty",
    "group": "Alerts",
    "index": 3,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "PAGERDUTY_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "ISE",
    "name": "ISE",
    "desc": "Collect endpoints and inventories from Cisco ISE",
    "long_desc": "Collect information about endpoints and inventories managed by Cisco ISE
 appliances and enrich endpoint inventories with user attributes and secure group tags
(SGT). Recommended usage with Cisco Secure Firewall connector for flow visibility.",
    "group": "Endpoints",
    "index": 1,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "LDAP",
      "LOG",
      "ALERT"
    ],
    "instance_spec": "ISE",
    "max_instances": 20,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "User Insights",
      "Inventory Enrichment",
      "Endpoint Insights",
      "Software Compliance Posture",
      "MDM Insights"
    ],
    "connector_count": 0,
    "group_order": 1
  },
  {
    "type": "MERAKI",
    "name": "Meraki",
    "desc": "Collect telemetry from Meraki firewalls",
    "long_desc": "Collect telemetry data from Meraki firewalls.",
    "group": "Flow Ingest",
    "index": 5,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
```

```
    "limit_per_appliance": 1,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "SERVICENOW",
    "name": "ServiceNow",
    "desc": "Collect ServiceNow CMDB records for inventories",
   "long_desc": "Collect CMDB information and service records from ServiceNow instance and
enriches endpoint inventories with cmdb attributes.",
    "group": "Inventory Enrichment",
    "index": 1,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "SERVICENOW",
      "SYNC_INTERVAL",
      "LOG",
      "ALERT"
    ],
    "instance_spec": "SERVICENOW",
    "max_instances": 10,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "User Insights",
      "Inventory Enrichment",
      "Endpoint Insights",
      "Software Compliance Posture"
    ],
    "connector_count": 0,
    "group_order": 2
  },
  {
    "type": "ERSPAN",
    "name": "ERSPAN",
    "desc": "Collect ERSPAN traffic",
    "long_desc": "",
    "group": "Flow Ingest",
    "index": 7,
    "appliance_type": "tetration_erspan",
    "state": "enabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 24,
    "limit_per_cluster": 450,
    "config": [],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
```

```
    "capabilities": [],
    "connector_count": 3,
    "group_order": 0
  },
  {
    "type": "AWS_CONNECTOR",
    "name": "AWS",
    "desc": "AWS Connector",
    "long_desc": "",
    "group": "Cloud",
    "index": 0,
    "appliance_type": "tetration_internal",
    "state": "disabled",
    "limit_per_appliance": 5000,
    "limit_per_rootscope": 5000,
    "limit_per_cluster": 100000,
    "config": [
      "AWS_CONNECTOR"
    ],
    "max_instances": 0,
    "noteworthy": true,
    "pre_release_tag": "BETA",
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Segmentation",
      "Managed K8s",
      "Inventory Enrichment"
    ],
    "connector_count": 0,
    "group_order": 5
  },
  {
    "type": "AZURE_CONNECTOR",
    "name": "Azure",
    "desc": "Azure Connector",
    "long_desc": "",
    "group": "Cloud",
    "index": 1,
    "appliance_type": "tetration_internal",
    "state": "disabled",
    "limit_per_appliance": 5000,
    "limit_per_rootscope": 5000,
    "limit_per_cluster": 100000,
    "config": [
      "AZURE_CONNECTOR"
    ],
    "max_instances": 0,
    "noteworthy": true,
    "pre_release_tag": "BETA",
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Segmentation",
      "Managed K8s",
      "Inventory Enrichment"
    ],
    "connector_count": 0,
    "group_order": 5
  },
  {
    "type": "GCP_CONNECTOR",
    "name": "GCP",
    "desc": "GCP Connector",
```

```
  "long_desc": "",
  "group": "Cloud",
  "index": 2,
  "appliance_type": "tetration_internal",
  "state": "disabled",
  "limit_per_appliance": 5000,
  "limit_per_rootscope": 5000,
  "limit_per_cluster": 100000,
  "config": [
    "GCP_CONNECTOR"
  ],
  "max_instances": 0,
  "noteworthy": true,
  "pre_release_tag": "BETA",
  "hidden": false,
  "capabilities": [
    "Managed K8s"
  ],
  "connector_count": 0,
  "group_order": 5
}]
```

API to Delete a Connector

This endpoint deletes the given Connector.

```
DELETE /openapi/v1/connectors/<id>
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID

Response object: Returns the status of the deleted Connector.

Sample response

```
resp = restclient.delete('/connectors/63c12e316419d0131767e21c')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
 {
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API to Get a Connector by ID

This endpoint gets the Connector with given ID.

```
GET /openapi/v1/connectors/<id>
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID

Response object: Returns the Connector with given ID.

Sample response

```
resp = restclient.get('/connectors/63c12e316419d0131767e21b')
 if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
{
    "name": "ERSPAN",
    "updated_at": 0,
    "created_at": 1673604657,
    "appliance_id": "63c1272039042a1c0ddd3e20",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "type": "ERSPAN",
    "ip_bindings": [],
    "internal": false,
    "id": "63c12e316419d0131767e21b"
 }
```

API to Rename a Connector

This endpoint renames the Connector with given ID.

```
PUT /openapi/v1/connectors/<id>
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID
name	string	Specify the new name of the Connector

Response object: Returns the Connector with given ID and the new name.

Sample response

```
req_payload = {
    "name": "ERSPAN2",
  }
  resp = restclient.put('/ext_appliances/63c12e316419d0131767e21b',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "name": "ERSPAN2",
    "updated_at": 0,
```

```
    "created_at": 1673604657,
    "appliance_id": "63c1272039042a1c0ddd3e20",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "type": "ERSPAN",
    "ip_bindings": [],
    "internal": false,
    "id": "63c12e316419d0131767e21b"
  }
```

API to Get the Connector Info with Details

This endpoint gets the Connector info with details.

```
GET /openapi/v1/connectors/cards/<type>?root_scope_id=<root_scope_id>
```

where <root_scope_id> is the root_scope_id that can be obtained from the Get scopes, on page 59 API. In request payload, <type> is a string to decide Connector type.

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Specify the root scope
type	string	Specify the Connector type, the value can be either "NETFLOW", "NETSCALER" "F5" "AWS" "ANYCONNECT" "ASA" "SLACK" "KINESIS" "SYSLOG" "EMAIL" "MERAKI" "PAGERDUTY" "ISE" "SERVICENOW" "ERSPAN" "AWS_CONNECTOR" "AZURE_CONNECTOR" "GCP_CONNECTOR"

Response object: Returns the Connectors under given scope.

Sample response

```
resp = restclient.get('/connectors/cards/NETFLOW?root_scope_id=63bf8d2f497d4f7287dbd335')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "type": "NETFLOW",
    "name": "NetFlow",
    "desc": "Collect telemetry from network devices",
    "long_desc": "Collect NetFlow V9 and/or IPFIX telemetry from network devices such as
routers and switches.",
    "group": "Flow Ingest",
    "index": 0,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
```

```
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility"
    ],
    "connector_count": 0,
    "info": {
      "help": "NetFlow connector collects telemetry data from various network devices (such
 as routers, switches).<br> It supports ingest of telemetry data in IPFIX and NetFlow V9
protocols. This connector can be used to discover inventory as it provides a network context.
 The connector helps convert data from flow exports and send them securely as Tetration
Flow records into an instance of Tetration. <br><br><b> Connector Alerts: </b><br> When
Connector Alerts are enabled, you may receive the following alerts: <br>   1. NetFlow
 Connector is down (due to missing heartbeats). <br>   2. Informational alert on high
 CPU/Memory usage."
    },
    "group_order": 0
  }
```


API to Get Connectors

This endpoint returns all Connectors for a given Appliance.

```
GET
/openapi/v1/connectors?root_scope_id=<root_scope_id>&appliance_id=<appliance_id>&type=<type>&state=<state>
```

where <root_scope_id> is the root_scope_id that can be obtained from the Get scopes, on page 59 API, <appliance_id> is the appliance_id that can be obtained from API to Get List of Appliances, on page 174, <type> is a string to decide Connector type that can be obtained from API to Get Appliance Schema, on page 184 "connectors" field, <state> is a filter for connectors state.

Parameters: The request URL contains the following parameters

Name	Type	Description
root_scope_id	string	Specify the root scope
appliance_id	string	Specify the Appliance ID
type	string	Specify the Connector type, the value can be either “NETFLOW”, “NETSCALER” “F5” “AWS” “ANYCONNECT” “ASA” “SLACK” “KINESIS” “SYSLOG” “EMAIL” “MERAKI” “PAGERDUTY” “ISE” “SERVICENOW” “ERSPAN” “AWS_CONNECTOR” “AZURE_CONNECTOR” “GCP_CONNECTOR”

Name	Type	Description
state	string	Filter the Connectors state, the value can be either “ENABLED” “DISABLED” or “UNAVAILABLE”

Response object: Returns all Connectors for given Appliance.

Sample response

```
resp =
restclient.get('root_scope_id=63bf8d2f497d4f7287dbd335&appliance_id=63c1272039042a1c0ddd3e20&type=ERSPAN&state=ENABLED')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
[
    {
      "name": "ERSPAN",
      "updated_at": 0,
      "created_at": 1673604657,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "type": "ERSPAN",
      "ip_bindings": [],
      "state": "enabled",
      "internal": false,
      "id": "63c12e316419d0131767e21b"
    },
    {
      "name": "ERSPAN",
      "updated_at": 0,
      "created_at": 1673604657,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "type": "ERSPAN",
      "ip_bindings": [],
      "state": "enabled",
      "internal": false,
      "id": "63c12e316419d0131767e21c"
    },
    {
      "name": "ERSPAN",
      "updated_at": 0,
      "created_at": 1673604657,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "type": "ERSPAN",
      "ip_bindings": [],
      "state": "enabled",
      "internal": false,
      "id": "63c12e316419d0131767e21d"
    }
  ]
```

API to Create a Connector

This endpoint creates a Connector for given Appliance.

```
POST /openapi/v1/connectors
```

In request payload, <root_scope_id> is the root_scope_id that can be obtained from the Get scopes, on page 59 API, <appliance_id> is the appliance_id that can be obtained from API to Get List of Appliances, on page 174, <type> is a string to decide Connector type that can be obtained from API to Get Appliance Schema, on page 184 "connectors" field.

Parameters: The request URL contains the following parameters

Name	Type	Description
name	string	Specify the name
root_scope_id	string	Specify the root scope
appliance_id	string	Specify the Appliance ID
type	string	Specify the Connector type, the value can be either “NETFLOW”, “NETSCALER” “F5” “AWS” “ANYCONNECT” “ASA” “SLACK” “KINESIS” “SYSLOG” “EMAIL” “MERAKI” “PAGERDUTY” “ISE” “SERVICENOW” “ERSPAN” “AWS_CONNECTOR” “AZURE_CONNECTOR” “GCP_CONNECTOR”
version	string	Specify the version

Response object: Returns the created connector.

Sample response

```
req_payload = {
    "root_scope_id": "63c41ff2497d4f5f5be73662",
    "appliance_id": "63c6ef42bca44e2b5e729191",
    "type": "NETFLOW",
    "name": "netflowtest",
    "version": "1.1.1"
  }
  resp = restclient.post('/connectors', json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "name": "netflowtest",
    "updated_at": 0,
    "created_at": 1674187875,
    "appliance_id": "63c6ef42bca44e2b5e729191",
    "root_scope_id": "63c41ff2497d4f5f5be73662",
    "vrf_id": 1,
```

```
    "type": "NETFLOW",
    "ip_bindings": [],
    "sources": [],
    "internal": false,
    "id": "63ca14631a49bd4c0e0cefa2"
  }
```

API to Get the Configs on Connector Config Type

This endpoint gets the configs of the Connector with given ID.

```
GET /openapi/v1/connectors/<id>/config?config_type=<config_type>
```

where <id> is the id that can be obtained from the API to Get List of Appliances, on page 174. <config_type> is the "valid_config" that can be obtained from API to Get Appliance Schema, on page 184.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID
config_type	string	Specify the config type. Refer to API to Get the Schema of Config, on page 171 for all possible values listed under Description

Response object: Returns the configs with given Connector ID and config type.

Sample response

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/config?config_type=LOG')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
 [{
    "mode": "TEST_AND_APPLY",
    "name": "Log instance 2/1/23 22:29",
    "root_scope_id": "63d98f45497d4f53005b24fa",
    "vrf_id": 1,
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "deleted": false,
    "type": "LOG",
    "state": "COMMIT",
    "error_code": "NO_ERROR",
    "error_text": "",
    "attempts": 1,
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    },
    "push_to_dio_at": 1675319360,
    "created_at": 1675319360,
```

```
    "updated_at": 1675319364,
    "discovery_completed_at": 0,
    "committed_at": 1675319364,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63db5840f029813659f9fcf5"
  }]
```

API to Add a New Config to Connector

This endpoint adds a new config to the Connector with given ID

```
POST /openapi/v1/connectors/<id>/config
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200. <config_type> is the "valid_config" that can be obtained from API to Get Appliance Schema, on page 184. To obtain <config> schema, select one of the "config" from API to Get All Types of Connectors, on page 189 response, apply the "config" to API to Get the Schema of Config, on page 171.

Parameters: The request URL contains the following parameters

Name	Type	Description
name	string	Specify the config name
type	string	Specify the config type. Refer to API to Get the Schema of Config, on page 171 for all possible values listed under Description
config	set	Provide the filled config schema in JSON format

Response object: Returns the updated config.

Sample response

```
req_payload = {
    "name": "Log instance 2/1/23 22:29",
    "type": "LOG",
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    }
  }
  resp = restclient.post('/connectors/63db5418e6ee1167a4c0986c/config',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "mode": "TEST_AND_APPLY",
    "name": "Log instance 2/1/23 11:29",
    "root_scope_id": "63d98f45497d4f53005b24fa",
    "vrf_id": 1,
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "deleted": false,
    "type": "LOG",
    "state": "PENDING",
    "attempts": 0,
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    },
    "push_to_dio_at": 0,
    "created_at": 1675322272,
    "updated_at": 1675322272,
    "discovery_completed_at": 0,
    "committed_at": 0,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63db63a0f029813659f9fcf7"
  }
```


API to Delete a Config

This endpoint deletes a config from given Connector.

```
DELETE /openapi/v1/connectors/<id>/config/<config_id>
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200, <config_id> is the id that can be obtained from API to Get the Configs on Connector Config Type, on page 203.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID
config_id	string	Specify the config ID

Response object: Returns the status of the config deleted for given Connector.

Sample response

```
resp =
restclient.delete('/connectors/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response


```
{
    "status": 200,
```

```
  "code": 1000,
  "message": "deleted"
}
```

-

API to Get the Config

This endpoint gets the config with given ID

```
GET /openapi/v1/connectors/<id>/config/<config_id>
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200, <config_id> is the id that can be obtained from API to Get the Configs on Connector Config Type, on page 203.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Appliance ID
config_id	string	Specify the config ID

Response object: Returns the config with given ID.

Sample response

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/config/63db5840f029813659f9fcf5')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "mode": "TEST_AND_APPLY",
    "name": "Log instance 2/1/23 22:29",
    "root_scope_id": "63d98f45497d4f53005b24fa",
    "vrf_id": 1,
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "deleted": false,
    "type": "LOG",
    "state": "COMMIT",
    "error_code": "NO_ERROR",
    "error_text": "",
    "attempts": 1,
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    },
    "push_to_dio_at": 1675319360,
    "created_at": 1675319360,
    "updated_at": 1675319364,
    "discovery_completed_at": 0,
    "committed_at": 1675319364,
    "delete_at": 0,
```

```
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63db5840f029813659f9fcf5"
  }
```

API to List Troubleshooting Commands Available for Connector

This endpoint returns the list of troubleshooting commands available for given Connector.

```
GET /openapi/v1/connectors/<id>/commands/available
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID

Response object: Returns the list of troubleshooting commands available for given Connector.

Sample response

```
resp = restclient.get('/connectors/63c6f2701a49bd2bb0696cab/commands/available')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
[
    {
      "type": "LIST_SERVICE_FILES",
      "name": "List a directory in a service"
    },
    {
      "type": "CONTROLLER_PROFILING",
      "name": "Collect controller profile"
    },
    {
      "type": "SHOW_LOG",
      "name": "Show logs"
    },
    {
      "type": "SHOW_SERVICE_LOG",
      "name": "Show service logs"
    },
    {
      "type": "RESTART_CONNECTOR_CONTAINER",
      "name": "Restart the connector docker container"
    },
    {
      "type": "SHOW_SUPERVISOR_COMMANDS",
      "name": "Execute supervisorctl command"
    },
    {
      "type": "CONNECTOR_ALERT_INTERVAL_CONNECTOR",
      "name": "Override connector alert interval for Connector"
    },
    {
      "type": "SERVICE_PROFILING",
      "name": "Collect connector profile"
```

```
    },
    {
      "type": "SNAPSHOT_CONNECTOR",
      "name": "Collect Snapshot from a connector"
    },
    {
      "type": "PACKET_CAPTURE",
      "name": "Packet capture"
    },
    {
      "type": "NETWORK_CONNECTIVITY_COMMANDS",
      "name": "Test network connectivity"
    },
    {
      "type": "SHOW_SERVICE_RUNNING_CONF",
      "name": "Show running configuration of a service"
    },
    {
      "type": "RESTART_CONNECTOR_SERVICE",
      "name": "Restart the connector service"
    },
    {
      "type": "SHOW_SYS_COMMANDS",
      "name": "Execute system command"
    }
  ]
```

API to List Troubleshooting Commands

This endpoint returns the list of troubleshooting commands that are enabled for given Connector.

```
GET /openapi/v1/connectors/<id>/commands
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID

Response object: Returns the list of troubleshooting commands that are enabled for given Connector.

Sample response

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/commands')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
[
    {
      "appliance_id": "63dad690e6ee1131f255e985",
      "connector_id": "63db5418e6ee1167a4c0986c",
      "service_id": "63db5418e6ee1167a4c0986d",
      "state": "success",
      "level": "SERVICE",
      "command": "SHOW_LOG",
      "arg_string": "",
      "args": {
        "pattern": "info"
      },
```

```
      "tailed": false,
      "rc": 0,
      "push_to_dio_at": 1675319615,
      "attempts": 1,
      "error_code": "NO_ERROR",
      "error_text": "",
      "deleted": false,
      "deleted_at": 0,
      "created_at": 1675319613,
      "total_chunk": 0,
      "id": "63db593df029813659f9fcf6"
    }
  ]
```

API to Create a Troubleshooting Command

This endpoint creates a troubleshooting command available for given Connector.

```
POST /openapi/v1/connectors/<id>/commands
```

here <id> is the id that can be obtained from the API to Get Connectors, on page 200. In request payload, <command> is a troubleshooting command type that can be obtained from API to List Troubleshooting Commands Available for Connector, on page 207. <arguments> is a filled JSON object of the command schema, which can be obtained from API to Get the Schema of Troubleshooting Commands, on page 173.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID
command	string	Specify the command type
arguments	set	Provide the filled command schema in JSON format

Response object: Returns the troubleshooting command created for given Appliance.

Sample response

```
req_payload = {
    "command": "SHOW_LOG",
    "arguments": {
      "pattern": "info"
    }
  }
  resp = restclient.post('/connectors/63db5418e6ee1167a4c0986c/commands',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "state": "pending",
    "level": "SERVICE",
    "command": "SHOW_LOG",
    "args": {
      "pattern": "info"
```

```
    },
    "tailed": false,
    "rc": 0,
    "push_to_dio_at": 0,
    "attempts": 0,
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1675319613,
    "total_chunk": 0,
    "id": "63db593df029813659f9fcf6"
  }
```

API to Delete a Troubleshooting Command

This endpoint deletes a troubleshooting command available for given Connector.

```
DELETE /openapi/v1/connectors/<id>/commands/<command_id>
```

where <id> is id that can be obtained from the API to Get Connectors, on page 200, <command_id> is the id that can be obtained from API to List Troubleshooting Commands, on page 186.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID
command_id	string	Specify the command ID

Response object: Returns the status of the troubleshooting command deleted for given Connector.

Sample response

```
resp =
restclient.delete('/connectors/63c12e316419d0131767e21c/commands/63c10a0039042a6aee1b008c')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response

```
{
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API to Return a Troubleshooting Command

This endpoint returns the selected troubleshooting command for a given connector.

```
GET /openapi/v1/connectors/<id>/commands/<command_id>
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200, <command_id> is the id that can be obtained from API to List Troubleshooting Commands, on page 186.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID

Name	Type	Description
command_id	string	Specify the command ID

Response object: Returns the selected troubleshooting command for a given connector.

Sample response

```
resp =
restclient.get('/connectors/63db5418e6ee1167a4c0986c/commands/63db593df029813659f9fcf6')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Sample response


```
{
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "state": "success",
    "level": "SERVICE",
    "command": "SHOW_LOG",
    "arg_string": "",
    "args": {
      "pattern": "info"
    },
    "tailed": false,
    "rc": 0,
    "push_to_dio_at": 1675319615,
    "attempts": 1,
    "error_code": "NO_ERROR",
    "error_text": "",
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1675319613,
    "total_chunk": 0,
    "id": "63db593df029813659f9fcf6"
  }
```


API to Download the Output of the Connector Command as a File

This endpoint downloads the output of the command as a file.

```
GET /openapi/v1/connectors/<id>/commands/{command_id}/download
```

where <id> is the id that can be obtained from the API to Get Connectors, on page 200, <command_id> is the id that can be obtained from API to List Troubleshooting Commands, on page 186. Not all commands have downloadable output, check the command schema that is provided by API to Get the Schema of Troubleshooting Commands, on page 173, where "output_type": "FILE" indicates it has downloadable content and "output_ext" tells the file type.

Parameters: The request URL contains the following parameters

Name	Type	Description
id	string	Specify the Connector ID
command_id	string	Specify the command ID

Response object: Returns the selected troubleshooting command for a given connector.

Sample response

```
resp = restclient.download('downloadFile',
'/connectors/63c6ef42bca44e2b5e729191/commands/63cace941a49bd4c0e0cf45a/download')
```