

# Policy Enforcer API Developer Guide

Published  
2022-09-01

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*Policy Enforcer API Developer Guide*

Copyright © 2022 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

## YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

## END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement (“EULA”) posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

# Table of Contents

1

## Configuring and Using Controller API

Understanding Controller API | 6

Configuring Controller API | 7

Using Controller API | 10

Tenant API Usage Examples | 10

Site API Usage Examples | 12

Policy Enforcement Group API Usage Examples | 16

Threat Policy API Usage Examples | 20

Custom Feed API Usage Examples | 24

Geo IP API Usage Examples | 32

Log API Usage Examples | 34

2

## Configuring and Using Connector API

Understanding the Policy Enforcer Connector API | 37

Using the Policy Enforcer Connector API | 39

Get Connector Type Details | 40

Modify Connector Server's Environment | 42

Get Connector Configuration Details | 43

Instantiate a Connector | 46

Get Connector Instance Details by ID | 48

Modify Connector Instance | 49

Delete a Connector Instance | 52

Get the Connector and its Controller Status | 52

Get All Enforcement Point Devices | 52

Get All Enforcement Subnets | 54

Retrieve Connector Log Files | 54

Block the Infected Host Threat | 55

Get All Infected Host Threats | 56

Block the DDoS Attack Threat | 57

Get a List of DDoS Threats	58
Get IP Addresses of the Endpoints	59
Get All Tag Names and Their Values	60
Register a Listener	61
Get All Registered Listeners	62
Get Details of a Listener by ID	63
Deregister a Listener by ID	64
Get a List of Controller Contexts	65
Get More Information About a Controller	67

# 1

CHAPTER

## Configuring and Using Controller API

---

Understanding Controller API | 6

Configuring Controller API | 7

Using Controller API | 10

---

# Understanding Controller API

The Controller API provides advanced threat prevention policies that you can apply on your network security devices and monitor them for attacks. You can use this API to gather and aggregate threat information from multiple locations and devices, both physical and virtual, as well as from third-party solutions. You can use this information to assess and manage threats on your network.

The Controller API is a pivotal component of Policy Enforcer. Policy Enforcer provides centralized, integrated threat management of all your security devices (both physical and virtual). For more information on Policy Enforcer, see [Policy Enforcer Administration Guide](#).

The following is a list of functionalities that you can perform using the various APIs defined under the Controller API:

- Configure Policy Enforcer settings using the Config API.
- Create secure fabrics/sites using the Site API.
- Create feed sources/Juniper ATP Cloud realms using the Feed Source API.
- Create policy enforcement groups using the Policy Enforcement Group API.
- Create threat prevention policies using the Threat Policy API.
- Create custom feeds using the Custom Feed API.
- Create Geo IP policies using the Geo IP API.
- Retrieve all log files in zip format using the Logs API.

You can also perform these activities using the Policy Enforcer UI. For more information, see [Policy Enforcer Administration Guide](#).

## RELATED DOCUMENTATION

[Configuring Controller API | 7](#)

[Using Controller API | 10](#)

# Configuring Controller API

You must perform some initial configuration and setup activities through the Policy Enforcer UI before you begin using the Controller API.

1. Log into the Policy Enforcer UI and configure the threat prevention type. You can configure Policy Enforcer to run in the following modes based on the threat prevention type you select:

- Cloud only or cloud feeds mode
- ATP Cloud mode
- ATP Cloud with Policy Enforcer mode

For more information on configuring threat prevention types, see [Policy Enforcer Administration Guide](#).

2. Invoke the Config API with HTTP basic authentication using your Policy Enforcer server SSH user credentials, as shown in the following example:

```
POST <context>/api/v2/controller/configs
    Content-Type: application/json
    Authorization: Basic <base 64 encoded
        (<ssh username of Policy Enforcer>:<ssh password of Policy Enforcer>)>
configs: {cloudOnly: false, purge_settings: {enable_purge: true, purge_history:
365}, version: "21.2R1-1227",...}

cloudOnly: false
customOnly: false
"ems": {"url": "https://192.168.250.3",
"username": "pe_user",
"password": "pe-root-pass_pe",
"SD_app_sub_version": null,"SD_app_version": "20.3","SD_release_type":
"R1","certificateMode": false,"certfile_content": null,"keyfile_content":
null}

certfile: null
certificateMode: false
keyfile: null
url: "https://10.204.253.148"
username: "pe_user"
```

```

purge_settings: {enable_purge: true, purge_history: 365}

enable_purge: true
  purge_history: 365

sdsn: true
sudoUser: "root"
syslog: {}
version: "21.2R1-1227"

```

**NOTE:**

- You must use only https for the ems URL.
- Ensure that you are using the correct Policy Enforcer username and password that successfully authenticate Security Director. The password should have \_pe as suffix.
- The REST API username and password are the new credentials for the REST APIs.

Based on the mode you have selected in the Policy Enforcer UI, you can specify the configuration using the Config API as follows:

- a. **Default mode**—You can configure Policy Enforcer in the Default mode as follows:

```

"configs": {
  "sdsn": true
  "cloudonly": false
  "customOnly": true
}

```

- b. **Cloud only or cloud feeds mode**—You can configure Policy Enforcer in the cloud only or cloud feeds mode as follows:

```

"configs": {
  "sdsn": true
  "cloudonly": true
}

```



```
"customOnly": false
}
```

- c. **ATP Cloud mode**—You can configure Policy Enforcer in the Juniper ATP Cloud mode as follows:

```
"configs": {
  "sdsn": false
  "cloudonly": false
  "customOnly": false
}
```

- d. **ATP Cloud with Policy Enforcer mode**—You can configure Policy Enforcer in the Juniper ATP Cloud with Policy Enforcer mode as follows:

```
"configs": {
  "sdsn": true
  "cloudonly": false
  "customOnly": false
}
```

**NOTE:** Ensure that the values of **sdsn** and **cloudonly** reflect the mode you have selected in the Policy Enforcer UI.

3. You can also use your REST API user credentials for HTTP basic authentication to make any Policy Enforcer REST API calls. To do so, you must first create your REST API username and password. You can use any value as the username and password, for example, **admin/admin** or **abcd/wxyz**.

## RELATED DOCUMENTATION

[Understanding Controller API | 6](#)

[Using Controller API | 10](#)

# Using Controller API

## IN THIS SECTION

- Tenant API Usage Examples | 10
- Site API Usage Examples | 12
- Policy Enforcement Group API Usage Examples | 16
- Threat Policy API Usage Examples | 20
- Custom Feed API Usage Examples | 24
- Geo IP API Usage Examples | 32
- Log API Usage Examples | 34

The following sections provide usage examples for the various APIs defined in the Controller API:

**NOTE:** For usage examples of the Config API, see [“Configuring Controller API” on page 7](#).

## Tenant API Usage Examples

The following are usage examples for the Tenant API:

### Usage Example 1 - Creating a Tenant

```
POST: /api/v2/controller/tenants
Content-Type: application/json'
Accept: application/json'
Body:
{
  "tenant": {
    "name": "COKE",
    "lsysName": "root-logical-system",
```

```

    "vrfName": "coke"
  }
}

```

### Usage Example 2 - Retrieving a Tenant

```

GET: /api/v2/controller/tenants/"6802b803-a20a-42a1-aab2-69b21c5c9f21"
Accept: application/json
output:
{
  "tenant": {
    "updateTs": 0,
    "name": "COKE",
    "uri": "/api/v2/controller/tenants/6802b803-a20a-42a1-aab2-69b21c5c9f21",

    "vrfName": "coke",
    "lsysName": "root-logical-system",
    "id": "6802b803-a20a-42a1-aab2-69b21c5c9f21"
  }
}

```

### Usage Example 3 - Updating a Tenant

```

PUT: /api/v2/controller/tenants/"6802b803-a20a-42a1-aab2-69b21c5c9f21"
Accept: application/json
Content-Type: application/json
Body:
{
  "tenant": {
    "updateTs": 0,
    "name": "COKE",
    "uri": "/api/v2/controller/tenants/6802b803-a20a-42a1-aab2-69b21c5c9f21",

    "vrfName": "coke-soda",
    "lsysName": "root-logical-system",
    "id": "6802b803-a20a-42a1-aab2-69b21c5c9f21"
  }
}

```

```

    }
    Output:
  {
    "tenant": {
      "vrfName": "coke-soda",
      "name": "COKE",
      "uri": "/api/v2/controller/tenants/6802b803-a20a-42a1-aab2-69b21c5c9f21",

      "updateTs": 0,
      "lsysName": "root-logical-system",
      "id": "6802b803-a20a-42a1-aab2-69b21c5c9f21"
    }
  }
}

```

#### Usage Example 4 - Deleting a Tenant

```
DELETE: /api/v2/controller/tenants/6802b803-a20a-42a1-aab2-69b21c5c9f21
```

## Site API Usage Examples

The following are usage examples for the Site API:

**NOTE:** APIs must include the authorization header based on the RestFul API user created through the Configuring Controller API.

#### Usage Example 1 - Creating a site

```
POST <context>/api/v2/controller/sites
Content-Type: application/json
STATUS: 200
```

```

{
  "site": {
    "name": "stie_179",
    "domain": "Global",
    "description": "",
    "tenantId": "6802b803-a20a-42a1-aab2-69b21c5c9f21"
    "feedSourceId": "",
    "members": [
      {
        "deviceInfo": {
          "perimeterDevice": true,
        },
        "id": "29",
        "type": "DEVICE"
      },
      {
        "deviceInfo": {
          "perimeterDevice": true,
        },
        "id": "27",
        "type": "DEVICE"
      }
    ]
  }
}

```

where:

- **feedSourceId** is the Juniper ATP Cloud realm and its value is **NULL** during the POST operation. When a site is associated to realms, you can update the site with **feedSourceId**.

#### Usage Example 2 - Retrieving a site

```

GET <context>/api/v2/controller/sites/{siteId}
Content-Type: application/json
STATUS: 200

{
  "site": {

```

```

"description": "",
"domain": "SD domain name",
"tenantId": "6802b803-a20a-42a1-aab2-69b21c5c9f21"
"feedSourceId": "277540f6-e640-4306-b5fc-8be6c978ecc0",
"id": "535aad4d-2525-4fb8-9551-bb3d56cff48e",
"members": [
  {
    "deviceInfo": {
      "cluster": false,
      "description": null,
      "emsSdId": "262181",
      "enrollStatus": null,
      "feedSourceConfigStatus": "SET_SUCCESS",
      "initStatus": "SUCCESS",
      "ip": "10.92.83.217",
      "model": "srx550m",
      "name": "un-srx550m-02",
      "perimeterDevice": true,
      "serialNumber": "DA3917AK0018"
    },
    "id": "29",
    "type": "DEVICE"
  },
  {
    "deviceInfo": {
      "cluster": false,
      "description": null,
      "emsSdId": "262175",
      "enrollStatus": null,
      "feedSourceConfigStatus": "SET_SUCCESS",
      "initStatus": "SUCCESS",
      "ip": "10.92.82.179",
      "model": "VSRX",
      "name": "vsrx-srini-179-D100",
      "perimeterDevice": true,
      "serialNumber": "A9D70E39FF31"
    },
    "id": "27",
    "type": "DEVICE"
  }
],
"name": "stie_179",

```

```

        "updateTs": 1539189977,
        "uri": "/api/v2/controller/sites/535aad4d-2525-4fb8-9551-bb3d56cff48e"
    }
}

```

### Usage Example 3 - Updating a site based on siteId

```

PUT <context>/api/v2/controller/sites/{siteId}
Content-Type: application/json
STATUS: 200

{
  "site": {
    "name": "stie_179",
    "description": "",
    "domain": "Global",
    "tenantId": "6802b803-a20a-42a1-aab2-69b21c5c9f21"
    "feedSourceId": "277540f6-e640-4306-b5fc-8be6c978ecc0",
    "id": "535aad4d-2525-4fb8-9551-bb3d56cff48e",
    "members": [
      {
        "deviceInfo": {
          "perimeterDevice": true,
        },
        "id": "29",
        "type": "DEVICE"
      },
      {
        "deviceInfo": {
          "perimeterDevice": true,
        },
        "id": "27",
        "type": "DEVICE"
      }
    ]
  }
}

```

### Usage Example 4 - Deleting a site

```
DELETE <context>/api/v2/controller/sites/{siteId}
STATUS: 204
```

## Policy Enforcement Group API Usage Examples

The following are usage examples for the Policy Enforcement Group API:

**NOTE:** APIs must include the authorization header based on the RestFul API user created through the Configuring Controller API.

### Usage Example 1 - Creating a new Policy Enforcement Group

```
POST <context>/api/v2/controller/policyGroups
Content-Type: application/json
STATUS: 200

"policyGroup": {
  "name": "sunnyvale",
  "domain": "SD domain name",
  "feedSourceId": "uuid-realm-1234"
  "description": "sunnyvale user endpoints",
  "groupType": "IP",
  "sites": [{ "siteId": "uuid-111", "name": "bldg-A",
    "uri", "/api/v2/controller/Sites/uuid-111"},
    { "siteId": "uuid-222", "name": "bldg-B",
    "uri", "/api/v2/controller/Sites/uuid-222"},
    { "siteId": "uuid-333", "name": "bldg-6",
    "uri", "/api/v2/controller/Sites/uuid-333"}
  ],
  "addressGroups": ["192.0.2.0/24", "198.51.100.0-198.51.100.255",
```



```

    "203.0.113.0"]
  }

```

where:

- **sites** and **addressGroups** are mutually exclusive.
- The value of **addressGroups** can be a single IP, an IP range, or an IP subnet.
- If the value of **groupType** is **IP**, **addressGroups** are populated; if the value is **LOCATION**, sites are populated.

#### Usage Example 2 - Retrieving a specific policy enforcement group based on policyGroupId

```

GET <context>/api/v2/controller/policyGroups/{policyGroupId}
Content-Type: application/json
STATUS: 200
Location-based:
{
  "policyGroup": {
    "addresses": [],
    "createTs": 1539190061,
    "description": "",
    "domain": "",
    "groupType": "LOCATION",
    "id": "6b2f9d7e-2079-42b1-8806-40d5315e64bc",
    "name": "peg_site",
    "sites": [
      {
        "id": "535aad4d-2525-4fb8-9551-bb3d56cff48e",
        "name": "stie_179"
      }
    ],
    "updateTs": 0
  }
}
IP-based (if IP subnet is a part of connector):
{
  "policyGroup": {
    "addresses": [
      {

```

```

        "connectorInfo": {
            "endpointAddressSpace": {
                "name": "",
                "type": "Global"
            },
            "name": "forescout",
            "type": "forescout"
        },
        "subnet": "192.168.199.254/24",
        "subnetDescription": "fs_199",
        "subnetId": "f5ab56ab-5ed3-4f80-a1b2-5b511dbf0019",
        "type": "SUBNET"
    }
],
"createTs": 1539189913,
"description": "",
"domain": "",
"groupType": "IP",
"id": "c8de4f43-6fe7-4ee4-bdee-344b7cbb1b6c",
"name": "peg",
"sites": [],
"updateTs": 0
}
}

```

### Usage Example 3 - Updating a specific policy enforcement group based on policyGroupId

```

PUT <context>/api/v2/controller/policyGroups/{policyGroupId}
Content-Type: application/json
STATUS: 200

"policyGroup": {
    "name": "sunnyvale",
    "domain": "SD domain name",
    "feedSourceId": "uuid-realm-1234"
    "description": "sunnyvale user endpoints",
    "sites": [],
    "addressGroups": ["192.0.2.0/24", "198.51.100.0-198.51.100.255",

```

```

    "203.0.113.0" ]
  }

```

#### Usage Example 4 - Retrieving the updated policy enforcement group to check if the updates are present

```

GET <context>/api/v2/controller/policyGroups
Content-Type: application/json
STATUS: 200

"policyGroups": {
  "uri": "https://<host>/<context>/api/v2/controller/policyGroups",
  "total": 2,
  "policyGroup": [
    { "id": "uuid-1234",
      "uri": "https://<host>/<context>/api/v2/controller/policyGroups/uuid-1234",

      "name": "sunnyvale", "domain": "SD domain name",
      "description": "sunnyvale user endpoints", "feedSourceId",
        "uuid-realm-1234"
      "sites": [],
        "addressGroups":
          [ "192.0.2.0/24", "198.51.100.0-198.51.100.255", "203.0.113.0" ] },
    { "id": "uuid-1234",
      "uri": "https://<host>/<context>/api/v2/controller/policyGroups/uuid-1234",

      "name": "sunnyvale", "domain": "SD domain name",
      "description": "sunnyvale user endpoints", "feedSourceId",
        "uuid-realm-1235"
      "sites": [ { "siteId": "uuid-111", "name": "bldg-A", "uri",
                    "/api/v2/controller/Sites/uuid-111" },
                  { "siteId": "uuid-222", "name": "bldg-B", "uri",
                    "/api/v2/controller/Sites/uuid-222" },
                  { "siteId": "uuid-333", "name": "bldg-6", "uri",
                    "/api/v2/controller/Sites/uuid-333" }
                ],
        "addressGroups": [] },
    ]
  }

```

### Usage Example 5 - Deleting a policy enforcement group

```
DELETE <context>/api/v2/controller/policyGroups/{policyGroupId}
STATUS: 204
```

## Threat Policy API Usage Examples

The following are usage examples for the Threat Policy API:

**NOTE:** APIs must include the authorization header based on the RestFul API user created through the Configuring Controller API.

### Usage Example 1 - Creating a new Threat Policy

```
POST <context>/api/v2/controller/threatPolicys
Content-Type: application/json
STATUS: 200

"threatPolicy": {
  "name": "simplePolicy",
  "domain": "SD domain name",
  "description": "with all profiles",
  "profiles": [{
    "feedType": "CnC",
    "actions": [{ "threatLevelStart": "0", "threatLevelEnd": "4",
                  "action": "PERMIT" },
                { "threatLevelStart": "5", "threatLevelEnd": "7",
                  "action": "LOG" },
                { "threatLevelStart": "8", "threatLevelEnd": "9",
                  "action": "BLOCK_CLOSE", "redirectUrl": "",
                  "customMessage": "" }
              ], {
    "feedType": "INFECTED_HOST",
```

```

    "actions": [
      {
        "threatLevelStart": "",
        "threatLevelEnd": "",
        "action": "BLOCK_QUARANTINE",
        "quarantineVlanName": "v999"
      }
    ],
    {
      "feedType": "MALWARE", "malwareProfileName": "scanAll",
      "https": true, "actions": [{"threatLevelStart": "0",
        "threatLevelEnd": "6", "action": "PERMIT"},
        {"threatLevelStart": "7", "threatLevelEnd": "9",
        "action": "BLOCK_CLOSE", "redirectUrl": "",
        "customMessage": "call IT support"}]
    }, {
      "feedType": "SMTP", "attachmentProfileName": "scanAll",
      "actions": [{"threatLevelStart": "0", "threatLevelEnd": "6",
        "action": "PERMIT"},
        {"threatLevelStart": "7", "threatLevelEnd": "9",
        "action": "BLOCK_DROP"}]
    }
  ],
  "secondaryActions": ["LOG"],
  "policyGroups": [{"policyGroupId": "uu-123", "name": "peg1"},
    {"policyGroupId": "uu-456", "name": "peg2"}],
  "deployStatus": "DRAFT"
}

```

where:

- The value of *action* can be **PERMIT**, **LOG**, **BLOCK\_DROP**, **BLOCK\_CLOSE**, **BLOCK\_QUARANTINE**, **MONITOR**.
- The value of *secondaryAction* can be **LOG\_ALL**, **LOG\_BLOCKED**, or **NONE**.
- If you specify **MALWARE** as the *feedType*, SRX takes a single threat level threshold value, that is, it allows two actions – permit and block.
- If you specify **GEO\_IP** as the *feedType*, then the SRX Series device has no threshold and allows permit or block.
- For *deployStatus*, you do not have to specify the values **DRAFT**, **ANALYSIS\_PROGRESS**, **READY\_TO\_DEPLOY**, and **DEPLOYED** for POST and PUT operations.

### Usage Example 2 - Updating a threat policy

```

PUT <context>/api/v2/controller/threatPolicys/uuid-1234/emsData
Content-Type: application/json
STATUS: 200
STATUS: 500 (It can have following errors)
    "no PerimeterFirewall found based on PEG, skipping analysis"
    "ATP analysis policy: <xyz> has aamw/infected-host profile, no argon
capable device, skipping analysis"

"threatPolicy": {
    "name": "simplePolicy",
    "domain": "SD domain name",
    "description": "with all profiles",
    "profiles": [],
    "secondaryActions": ["LOG"],
    "policyGroups": [{"policyGroupId": "uu-123", "name": "peg1"},
{"policyGroupId": "uu-456", "name": "peg2"}],
    "deployStatus": "DRAFT",
    "emsAnalysisId": "uuid-policy-analysis",
    "emsPublishUpdateId": "publish-update-job-id"
}

```

### Usage Example 3 - Retrieving a specific threat policy based on threatPolicyId

```

GET <context>/api/v2/controller/threatPolicys/uuid-1234
Content-Type: application/json
STATUS: 200

"threatPolicy": {
    "id": "uuid-1234",
    "uri": "https://<host>/<context>/api/v2/controller/threatPolicys/uuid-1234",

    "name": "simplePolicy",
    "domain": "SD domain name",
    "description": "with all profiles",
    "profiles": [{
        "feedType": "CnC",
        "actions": [{"threatLevelStart": "0", "threatLevelEnd": "4",
                    "action": "PERMIT"}],
    }],
}

```

```

        {"threatLevelStart": "5", "threatLevelEnd": "7",
         "action": "LOG"},
        {"threatLevelStart": "8", "threatLevelEnd": "9",
         "action": "BLOCK_CLOSE",
         "redirectUrl": "", "customMessage": ""}]
    }, {
    "feedType": "INFECTED_HOST",
    "actions": [{"threatLevelStart": "0", "threatLevelEnd": "4",
                 "action": "PERMIT"},
                {"threatLevelStart": "8", "threatLevelEnd": "9",
                 "action": "BLOCK_QUARANTINE",
                 "quarantineVlanName": "911"}]
    }, {
    "feedType": "MALWARE", "malwareProfileName": "scanAll",
    "https": true, "actions": [{"threatLevelStart": "0",
                               "threatLevelEnd": "6", "action": "PERMIT"},
                               {"threatLevelStart": "7", "threatLevelEnd": "9",
                                "action": "BLOCK_CLOSE",
                                "redirectUrl": "", "customMessage": "call IT support"}]

    }, {
    "feedType": "SMTP", "attachmentProfileName": "scanAll",
    "actions": [{"threatLevelStart": "0", "threatLevelEnd": "6",
                 "action": "PERMIT"},
                {"threatLevelStart": "7", "threatLevelEnd": "9",
                 "action": "BLOCK_DROP"}]
    }
],
"secondaryActions": ["LOG"],
"policyGroups": [{"policyGroupId": "uu-123", "name": "peg1"},
                  {"policyGroupId": "uu-456", "name": "peg2"}],
"deployStatus": "DRAFT",
"deployDevices": [{"name": "device1", "deviceId": "uuid1234"}],
"skipDevices": [{"name": "device2", "deviceId": "uuid5678"}]
}

```

#### Usage Example 4 - Deleting a threat policy

```
DELETE <context>/api/v2/controller/threatPolicys/uuid-1234",
STATUS: 204
```

## Custom Feed API Usage Examples

The following are usage examples for the Custom Feed API:

**NOTE:** APIs must include the authorization header based on the RestFul API user created through the Configuring Controller API.

### Usage Example 1 - Creating a new CustomFeed with Local Files

```
POST <context>/api/v2/controller/customFeeds
Content-Type: application/json
Accept: application/json
STATUS: 200
Body - Version 1:
{
  "customFeed":
  {
    "feedType": "Dynamic-Address",    <= Can be
'Dynamic-Address/Allowlist/Blocklist'
    "domain": "Global",
    "name": "testda",
    "fileType": "Local",            <= Can be 'Local/Remote'
    "inputType": "ip",             <= Can be 'ip for Dynamic-Address and ip/url/domain
for Allowlist and Blocklist'
    "description": "",
    "content":
    [
      {
        "siteIds":
```



```

[
  "bc065f26-b081-43a8-bd37-e3f349cddbda" <= List of site ids based on the
creation of site in Secure Fabric page. See below to get site information.
],
"data":
[
  "1.1.1.1",
  "2.2.2.2" <= IP/URL/Domain list.
]
}
]
}
}

```

where:

- The value of *feedType* can be **Blocklist**, **Allowlist**, or **Dynamic-Address**.
- The value of *content* can be a list of IP addresses, an IP range, or a subnet for a **Blocklist**, **Allowlist** and, **Dynamic-Address**.
- The value of *inputType* can be an IP, URL or a domain for a **Blocklist**, **Allowlist** and, **Dynamic-Address**.

#### Usage Example 2- Creating a new CustomFeed with Infected-Host feedtype

```

POST <context>/api/v2/controller/customFeeds
Content-Type: application/json
Accept: application/json
STATUS: 200
Body - Version 2:
{
  "customFeed":
  {
    "feedType": "Infected-Hosts", <= Can be 'DDoS/Infected-Hosts'
    "domain": "Global",
    "name": "testih",
    "fileType": "Local", <= Can be 'Local/Remote'
    "inputType": "ip", <= Can only be 'ip for Infected-Hosts and DDoS'
    "description": "",
    "content":

```

```

[
  {
    "siteIds":
    [
      "bc065f26-b081-43a8-bd37-e3f349cddbda" <= List of site ids based on the
creation of site in Secure Fabric page. See below to get site information.
    ],
    "data":
    {
      "add": ["192.0.2.0", "198.51.100.0"] <= Will add new data.
    }
  }
]
}
}

```

where:

- The value of *feedType* is **Infected-Hosts**.
- The value of *content* can be a list of IP addresses.
- The value of *inputType* can be and an IP address.

### Usage Example 3- Creating a new Custom Feed with CC as feed type

```

POST
/api/v2/controller/customFeeds/CCCF1

Body:
{
  "customFeed": {
    "domain": "Global",
    "feedType": "CC",
    "name": "CCCF1",
    "fileType": "Local",
    "content": [
      {
        "siteIds": [
          "0feaf878-c8e6-45bb-aba3-8bf7c9ac83ae"

```

```

        ],
        "data": [
            {"value": "2.2.2.2", "threat_level": 2},
            {"value": "3.3.3.3", "threat_level": 10},
            {"value": "4.4.4.4", "threat_level": 10},
            {"value": "1.1.1.1", "threat_level": 3}
        ],
    }
],
"inputType": "ip",
"description": ""
}
}

```

If file upload UI should accept file with "," -> IP/URL,threat\_level

File contents:

2.2.2.2,3

4.4.4.4,5

#### Usage Example 4- Retrieving a specific custom feed based on CustomFeed Id

```
GET <context>/api/v2/controller/customFeeds/<name>
```

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

```
STATUS: 200
```

```
Response:
```

```

{
  "customFeed":
  {
    "username":null,
    "domain":"Global",
    "feedType":"Dynamic-Address",
    "name":"testda",
    "url":null,
    "fileType":"Local",
    "caCerts":null,
    "content":
    [
      {
        "siteIds":

```

```

    [
      "bc065f26-b081-43a8-bd37-e3f349cddbda"
    ],
    "data":
    [
      "1.1.1.1"
    ],
    "feedId": 1  <= This is the feedId that should be used during PUT
operation.
  }
],
"fail_count": 0,
"updateTs":1523460907,
"updateInterval":null,
"password":null,
"inputType":"ip",
"urlType":null,
"description":""
}
}

```

#### Usage Example 5- Retrieving a specific infected-host custom feed based on CustomFeed Id

```

GET <context>/api/v2/controller/customFeeds/<name>
Content-Type: application/json
STATUS: 200
Response:
"customFeed": {
  "id": "uuid-1234",
  "emsVersion":0,
  "createTs":1479328662,
  "emsAddressId":null,
  "updateTs":null
  "uri": "/api/v2/controller/customFeeds/uuid-1234",
  "name": "customIPs",
  "domain": "SD domain name",
  "description": "infected IPs",
  "feedType": "Infected-Hosts",
  "inputType": "ip",

```

```

"content": { "add": ["192.0.2.0", "198.51.100.0"],
             {"delete": ["198.51.100.255"]}
}

```

### Usage Example 6- Retrieving the list of custom feeds

```

GET <context>/api/v2/controller/customFeeds/
Content-Type: application/json
STATUS: 200

Response:
{
  "customFeeds":
  {
    "total":1,
    "customFeed":
    [
      {
        "updateTs": 1523460907,
        "name": "testda",
        "fileType": "Local",
        "content": [
          {
            "siteIds": [
              "bc065f26-b081-43a8-bd37-e3f349cdbcdbab"
            ]
          }
        ],
        "fail_count": 0,
        "createdByUser": 1,
        "feedType": "Dynamic-Address",
        "description": ""
      }
    ]
  }
}

```

### Usage Example 7- Updating a custom feed

```
PUT <context>/api/v2/controller/customFeeds/<name>
Content-Type: application/json
STATUS: 200
Body - Version 1:
{
  "customFeed":
  {
    "feedType": "Dynamic-Address",
    "domain": "Global",
    "name": "testda",
    "fileType": "Local",
    "inputType": "ip",
    "description": "",
    "content":
    [
      {
        "siteIds":
        [
          "bc065f26-b081-43a8-bd37-e3f349cddbda"
        ],
        "data":
        [
          "3.3.3.3",
          "1.1.1.1",
          "2.2.2.2"
        ],
        "feedId": 1  <= feedId should be retrieved from GET call. See below GET
API for details.
      }
    ]
  }
}
Body - Version 2:
{
  "customFeed":
  {
    "feedType": "Infected-Hosts",
    "domain": "Global",
    "name": "testih",
    "fileType": "Local",
    "inputType": "ip",
```

```

"description": "",
"content":
[
  {
    "siteIds":
    [
      "bc065f26-b081-43a8-bd37-e3f349cddbda"
    ],
    "data":
    {
      "add": ["192.0.2.1"],    <= Will add new data to existing list.
      "delete": ["192.0.2.0"] <= Will remove data from existing list.
    },
    "feedId": 2 <= feedId should be retrieved from GET call. See below GET
API for details.
  }
]
}
}
}

```

#### Usage Example 8- Deleting a custom feed

```

DELETE <context>/api/v2/controller/customFeeds/<name>
STATUS: 204

```

#### Usage Example 9- Creating Custom Feeds with Remote File Server

```

POST <context>/api/v2/controller/customFeeds
Body:
{
  "customFeed":
  {
    "feedType": "Dynamic-Address", <= Can be
'Dynamic-Address/Allowlist/Blocklist/Infected-Hosts/DDoS'
    "domain": "Global",
    "name": "testsda",

```

```

    "fileType": "Remote",
    "inputType": "ip",    <= Can be 'ip for Dynamic-Address/Infected-Hosts/DDoS
and ip/url/domain for Allowlist and Blocklist'
    "description": "",
    "content":
    [
      {
        "siteIds":
        [
          "bc065f26-b081-43a8-bd37-e3f349cddbda"
        ],
        "data": {}
      }
    ],
    "username": "super",
    "password": "123juniper",
    "updateInterval": "hourly", <= Can be hourly/monthly/yearly
    "url": "http://1.1.1.1/ip.list",
    "urlType": "http"    <= Can be http/https
    "caCerts": "un-esxi-01-1-vm29englabjunipernet.crt" <= If https
    "caCertsContent": "-----BEGIN
    [REDACTED]
    CERTIFICATE-----\r\n"
  }
}

```

## Geo IP API Usage Examples

The following are usage examples for Geo IP API:

**NOTE:** APIs must include the authorization header based on the RestFul API user created through the Configuring Controller API.

### Usage Example 1 - Creating a new Geo IP



```
POST <context>/api/v2/controller/geoIps
Content-Type: application/json
STATUS: 200
```

```
"geoIp": {
  "name": "asia",
  "domain": "SD domain name",
  "description": "all asia countries",
  "country": [CN, IN],
  "action": "BLOCK_INBOUND",
  "secondaryAction": "LOG"
}
```

where:

- The value of *action* can be **BLOCK\_INBOUND**, **BLOCK\_OUTBOUND**, or **BLOCK\_BOTH**.
- The value of *secondaryAction* can be **LOG** or **NONE**.

**NOTE:** The values for *action* and *secondaryAction* are only needed for SDSN.

### Usage Example 2 - Retrieving a specific Geo IP based on geoIpId

```
GET <context>/api/v2/controller/geoIps/{geoIpId}
Content-Type: application/json
STATUS: 200

"geoIp": {
  "id": "uuid-1234",
  "uri": "https://<host>/<context>/api/v2/controller/geoIps/uuid-1234",
  "name": "asia",
  "domain": "SD domain name",
  "description": "all asia countries",
  "country": [CN, IN],
  "action": "INBOUND",
  "secondaryAction": "LOG"
}
```

### Usage Example 3 - Retrieving the list of Geo IPs

```

GET <context>/api/v2/controller/geoIps
Content-Type: application/json
STATUS: 200

"geoIps": {
  "uri": "https://<host>/<context>/api/v2/controller/geoips",
  "total": 2,
  "geoip": [
    {"id": "uuid-1234", "name": "asia", "domain": "SD domain name",
      "description": "all asia countries", "country": [CN, IN],
      "action": "INBOUND", "secondaryAction": "LOG"},
    {"id": "uuid-1235", "name": "north korea", "domain": "SD domain name",
      "description": "some countries", "country": [KP],
      "action": "INBOUND", "secondaryAction": "LOG"}
  ]
}

```

### Usage Example 4 - Deleting Geo IP

```

DELETE <context>/api/v2/controller/geoIps/{geoIpId}
STATUS: 204

```

## Log API Usage Examples

The following is a usage example for Log API:

**NOTE:** APIs must include the authorization header based on the RestFul API user created through the Configuring Controller API.

### Usage Example - Retrieving all log files in zip format

```
GET <context>/api/v1/controller/logs  
STATUS: 200
```

## RELATED DOCUMENTATION

[Understanding Controller API | 6](#)

[Configuring Controller API | 7](#)

---

# 2

CHAPTER

## Configuring and Using Connector API

---

[Understanding the Policy Enforcer Connector API | 37](#)

[Using the Policy Enforcer Connector API | 39](#)

---

# Understanding the Policy Enforcer Connector API

The Policy Enforcer Connector framework supports integration of Policy Enforcer with other external third party systems for the following capabilities:

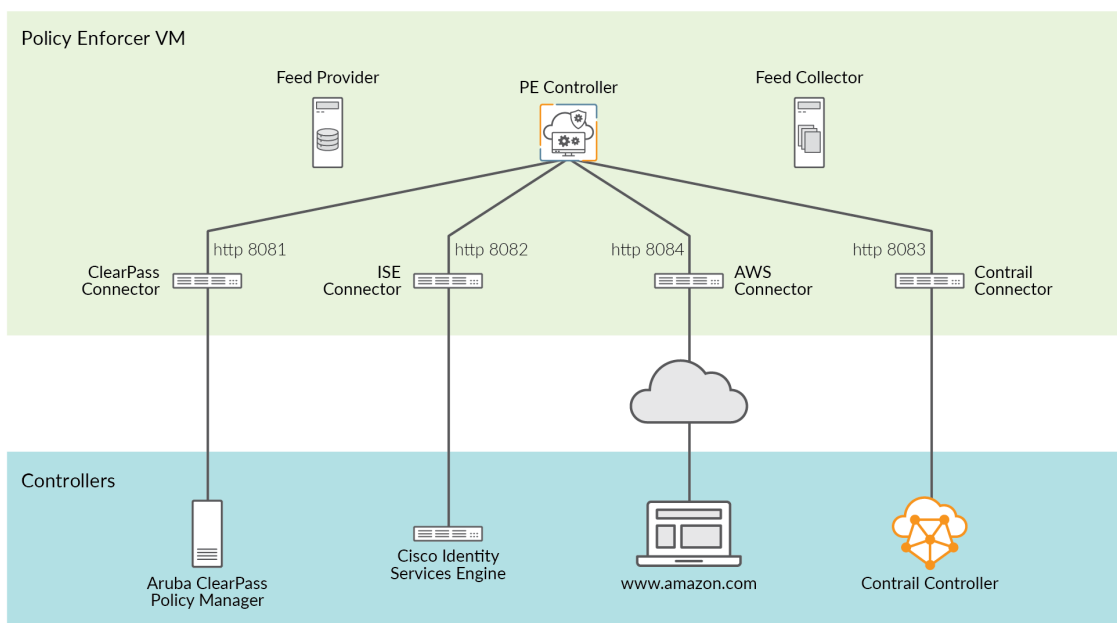
- Threat remediation of endpoints for both infected and DDOS attacks on a multivendor network.
- Application level micro segmentation advanced Layer 7 firewall security for workloads in public and private clouds, and for on-premises deployments.

Based on the integration requirement with the specific external system, the connector plugins can be implemented in adherence to the Connector Framework API specification. Currently, Policy Enforcer bundles the following plug-ins:

- Aruba ClearPass, Cisco Identity Services Engine (ISE), ForeScout CounterAct, Pulse Policy Secure for threat remediation use cases.
- Amazon Web Services (AWS), Microsoft Azure, VMWare NSX, and Juniper Contrail Cloud controllers for application Layer 7 advance firewall security.

Policy Enforcer provides a set of APIs and schemas to third party vendors to implement the plug-in accordingly. The plug-in can be deployed in a separate Virtual Machine (VM) and communicate with Policy Enforcer via the HTTP protocol after registering with Policy Enforcer, as shown in [Figure 1 on page 37](#).

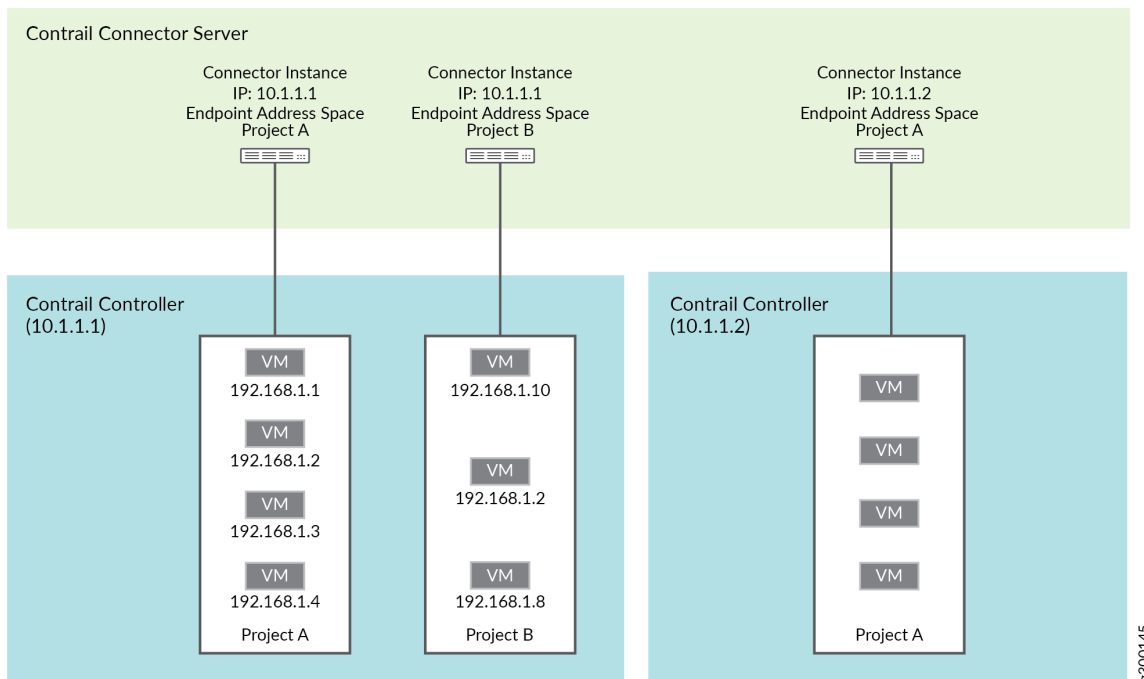
**Figure 1: Plug-in Communication with Policy Enforcer**



Configure a connector for third-party products to unify policy enforcement across all network elements. You must allocate a new port number when you create a new connector. For each connector, you must instantiate connector instances. These connector instances are similar to logical configurations within the connector servers. You require connector instances for every unique address space that you want to support.

For example, in Contrail system, Projects implement the unique address spaces for set of VMs, as shown in [Figure 2 on page 38](#). There could be overlapping address spaces between projects. However, you must instantiate connector instances and point them to these address spaces in your connector server. Instantiate a connector instance identified by the controller IP address (the address of the external controller system) and provide the name for an endpoint address space. The combination of the controller IP address and the endpoint address space name identifies the connector instance.

**Figure 2: Contrail Connector Server**



Inputs to instantiate a connector are received through the Policy Enforcer user interface (UI) and passed them to an appropriate connector instance through the POST API call. Use the POST /connector/instances API and specify the following parameters:

- Controller IP address
- Controller port number
- Username of the controller

- Password of the controller
- Endpoint address space information
  - Type—Specify Project for Contrail and VPC for AWS.
  - Name—Unique name for the specified Project or VPC.

Once you create a connector instance, the connector server allocates a unique connector ID. You can use this connector ID to initiate a GET call to retrieve the same information that you configured using the POST method.

#### RELATED DOCUMENTATION

| [Using the Policy Enforcer Connector API | 39](#)

## Using the Policy Enforcer Connector API

#### IN THIS SECTION

- [Get Connector Type Details | 40](#)
- [Modify Connector Server's Environment | 42](#)
- [Get Connector Configuration Details | 43](#)
- [Instantiate a Connector | 46](#)
- [Get Connector Instance Details by ID | 48](#)
- [Modify Connector Instance | 49](#)
- [Delete a Connector Instance | 52](#)
- [Get the Connector and its Controller Status | 52](#)
- [Get All Enforcement Point Devices | 52](#)
- [Get All Enforcement Subnets | 54](#)
- [Retrieve Connector Log Files | 54](#)
- [Block the Infected Host Threat | 55](#)
- [Get All Infected Host Threats | 56](#)
- [Block the DDoS Attack Threat | 57](#)
- [Get a List of DDoS Threats | 58](#)
- [Get IP Addresses of the Endpoints | 59](#)

- [Get All Tag Names and Their Values | 60](#)
- [Register a Listener | 61](#)
- [Get All Registered Listeners | 62](#)
- [Get Details of a Listener by ID | 63](#)
- [Deregister a Listener by ID | 64](#)
- [Get a List of Controller Contexts | 65](#)
- [Get More Information About a Controller | 67](#)

The following sections provide usage examples for the various APIs defined in the Plug-in/Adaptor Schema for External System Integration with PE API:

## Get Connector Type Details

Use this request to obtain information about the type of connector implemented by Policy Enforcer.

URI	/api/v2/connector
HTTP method	GET
Content-Type	application/json
Status	200  Returns the following information: <ul style="list-style-type: none"> <li>● Name identifying the type of connector. For example, HPClearPass, CiscoISE, and so on.</li> <li>● Information about additional settings supported by the connector</li> <li>● Indicates whether or not communication with a controller requires a context specifier to be provided in addition to the standard controller information, such as IP address, port, and credentials. For example, for AWS, a region name must be specified.</li> </ul>

### Usage Example

Body: None.



Response:

```
{
  "additionalSettings": [
    {
      "description": "Infected Host Security Group",
      "name": "SECURITY_GROUP_NAME",
      "required": true,
      "type": "string"
    },
    {
      "description": "SRX Identifier Tag",
      "name": "SRX_ID_TAG",
      "required": true,
      "type": "string"
    },
    {
      "description": "SRX Username",
      "name": "SRX_USERNAME",
      "required": true,
      "type": "string"
    },
    {
      "description": "SRX Authentication Key",
      "name": "PRIVATE_KEY_FILE",
      "required": true,
      "type": "file"
    }
  ],
  "capabilities": [
    {
      "name": "INFECTED_HOST_BLOCK"
    },
    {
      "name": "INFECTED_HOST_QUARANTINE"
    },
    {
      "name": "ENFORCEMENT_SUBNET_INFO"
    },
    {
      "name": "ENDPOINT_TAGGING_INFO"
    }
  ]
}
```

```

        "name": "ENFORCEMENT_POINT_INFO"
    }
],
"controllerContexts": true,
"type": "Amazon Web Services"
}

```

## Modify Connector Server's Environment

Use this request to update information about the environment of connector server.

URI	/api/v2/connector/env
HTTP method	PUT
Content-Type	application/json
Status	200—Indicates the connector environment information is successfully received.

### Usage Example

```

Request:
{
  "proxyServers": [
    {
      "type": "http",
      "ipAddress": "192.168.10.1",
      "port": 8080,
      "username": "proxyuser",
      "password": "juniper123"
    },
    {
      "type": "https",
      "ipAddress": "192.168.10.1",

```

```

        "port": 8443,
        "username": "proxyuser",
        "password": "juniper123"
    }
]
}

Response:
{
  "proxyServers": [
    {
      "type": "http",
      "ipAddress": "192.168.10.1",
      "port": 8080,
      "username": "proxyuser",
      "password": "juniper123"
    },
    {
      "type": "https",
      "ipAddress": "192.168.10.1",
      "port": 8443,
      "username": "proxyuser",
      "password": "juniper123"
    }
  ]
}

```

## Get Connector Configuration Details

Use this request to obtain the configuration information of all connectors.

URI	/api/v2/connector/instances
HTTP method	GET
Content-Type	application/json
Status	200—Returns a list of connector configurations.

## Usage Example

```

Body: None.
Response:
{
  "connectors": [
    {
      "additionalSettings": {
        "settings": [
          {
            "name": "SRX_USERNAME",
            "value": "root"
          },
          {
            "name": "SRX_ID_TAG",
            "value": "VSRX"
          },
          {
            "name": "PRIVATE_KEY_FILE",
            "value":
"https://127.0.0.1:8080/api/v2/controller/files/852af83b-0a6d-40c9-b397-4964de59ccc3"

          },
          {
            "name": "SECURITY_GROUP_NAME",
            "value": "aws_sg_block"
          }
        ],
        "total": 4
      },
      "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
      "controllerContext": "US West (N. California)",
      "controllerEndpointAddressSpace": {
        "name": "vpc-90e3f1f4_VPC_SDSN",
        "type": "Vpc"
      },
      "controllerIp": "aws.amazon.com",
      "controllerPort": 443,
      "controllerSecret": "1ggYYha0OCF0kG9qaJMG0Pgjad3rStvAeHgHmVc",
      "id": 1,
      "name": "aws_31",
      "uri": "http://127.0.0.1:8084/api/v2/connector/instances/1"
    }
  ]
}

```

```

    },
    {
      "additionalSettings": {
        "settings": [
          {
            "name": "SRX_USERNAME",
            "value": "root"
          },
          {
            "name": "SRX_ID_TAG",
            "value": "vSRX"
          },
          {
            "name": "PRIVATE_KEY_FILE",
            "value":
"https://127.0.0.1:8080/api/v2/controller/files/fc6aca5e-f3a7-4d9e-9bac-bf5fbfab4be3"

          },
          {
            "name": "SECURITY_GROUP_NAME",
            "value": "aws_sg_block"
          }
        ],
        "total": 4
      },
      "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
      "controllerContext": "US West (Oregon)",
      "controllerEndpointAddressSpace": {
        "name": "vpc-4bf15832_un-ore-vpc01",
        "type": "Vpc"
      },
      "controllerIp": "aws.amazon.com",
      "controllerPort": 443,
      "controllerSecret": "1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc",
      "id": 2,
      "name": "aws_31",
      "uri": "http://127.0.0.1:8084/api/v2/connector/instances/2"
    }
  ],
  "total": 2
}

```

## Instantiate a Connector

Use this request to instantiate a connector with the specified configuration.

URI	/api/v2/connector/instances
HTTP method	POST
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Indicates the the connector is successfully instantiated with the specified configuration.</li> <li>• 500—Unexpected error</li> </ul>

### Usage Example

Body:

```
{
  "name": "aws_31",
  "controllerIp": "aws.amazon.com",
  "controllerPort": 443,
  "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
  "controllerSecret": "1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc",
  "controllerContext": "US West (N. California)",
  "controllerEndpointAddressSpace": {
    "name": "vpc-90e3f1f4_VPC_SDSN",
    "type": "Vpc"
  },
  "additionalSettings": {
    "total": 4
    "settings": [
      {
        "name": "SRX_USERNAME",
        "value": "root"
      },
      {
        "name": "SRX_ID_TAG",
        "value": "VSRX"
      },
      {
        "name": "PRIVATE_KEY_FILE",
```

```

        "value":
"https://127.0.0.1:8080/api/v2/controller/files/852af83b-0a6d-40c9-b397-4964de59ccc3"

    },
    {
        "name": "SECURITY_GROUP_NAME",
        "value": "aws_sg_block"
    }
]
}
}

```

Response:

```

{
  "id": "1",
  "uri": "http://127.0.0.1:8084/api/v2/connector/instances/1",
  "name": "aws_31",
  "controllerIp": "aws.amazon.com",
  "controllerPort": 443,
  "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
  "controllerSecret": "lggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc",
  "controllerContext": "US West (N. California)",
  "controllerEndpointAddressSpace": {
    "name": "vpc-90e3f1f4_VPC_SDSN",
    "type": "Vpc"
  },
  "additionalSettings": {
    "total": 4,
    "settings": [
      {
        "name": "SRX_USERNAME",
        "value": "root"
      },
      {
        "name": "SRX_ID_TAG",
        "value": "VSRX"
      },
      {
        "name": "PRIVATE_KEY_FILE",
        "value":
"https://127.0.0.1:8080/api/v2/controller/files/852af83b-0a6d-40c9-b397-4964de59ccc3"

```

```

    },
    {
      "name": "SECURITY_GROUP_NAME",
      "value": "aws_sg_block"
    }
  ]
}
}

```

## Get Connector Instance Details by ID

Use this request to obtain configuration information for the specified connector instance.

URI	/api/v2/connector/instances/{connectorId}
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns connector configuration.</li> <li>• 500—Unexpected error</li> </ul>

### Usage Example

```

Body: None.
Response:
{
  "id": "1",
  "uri": "http://127.0.0.1:8084/api/v2/connector/instances/1",
  "name": "aws_31",
  "controllerIp": "aws.amazon.com",
  "controllerPort": 443,
  "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
  "controllerSecret": "1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc",
  "controllerContext": "US West (N. California)",

```



```

"controllerEndpointAddressSpace": {
  "name": "vpc-90e3f1f4_VPC_SDSN",
  "type": "Vpc"
},
"additionalSettings": {
  "total": 4,
  "settings": [
    {
      "name": "SRX_USERNAME",
      "value": "root"
    },
    {
      "name": "SRX_ID_TAG",
      "value": "VSRX"
    },
    {
      "name": "PRIVATE_KEY_FILE",
      "value":
"https://127.0.0.1:8080/api/v2/controller/files/852af83b-0a6d-40c9-b397-4964de59ccc3"
    }
  ],
  {
    "name": "SECURITY_GROUP_NAME",
    "value": "aws_sg_block"
  }
]
}
}

```

## Modify Connector Instance

Use this request to update the configuration for the specified connector instance.

URI	/api/v2/connector/instances/{connectorId}
HTTP method	PUT
Content-Type	application/json

Status

- 200—Connector configuration is updated successfully
- 500—Unexpected error

## Usage Example

Request:

```
{
  "id": "1",
  "name": "aws_31",
  "controllerIp": "aws.amazon.com",
  "controllerPort": 443,
  "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
  "controllerSecret": "1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc",
  "controllerContext": "US West (N. California)",
  "controllerEndpointAddressSpace": {
    "name": "vpc-90e3f1f4_VPC_SDSN",
    "type": "Vpc"
  },
  "additionalSettings": {
    "total": 4
    "settings": [
      {
        "name": "SRX_USERNAME",
        "value": "root"
      },
      {
        "name": "SRX_ID_TAG",
        "value": "VSRX"
      },
      {
        "name": "PRIVATE_KEY_FILE",
        "value":
          "https://127.0.0.1:8080/api/v2/controller/files/852af83b-0a6d-40c9-b397-4964de59ccc3"
      },
      {
        "name": "SECURITY_GROUP_NAME",
        "value": "aws_sg_block"
      }
    ]
  }
}
```

```

}

Response:
{
  "id": "1",
  "uri": "http://127.0.0.1:8084/api/v2/connector/instances/1",
  "name": "aws_31",
  "controllerIp": "aws.amazon.com",
  "controllerPort": 443,
  "controllerClientId": "AKIYXUTNS4UIQ5824BMQ",
  "controllerSecret": "1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc",
  "controllerContext": "US West (N. California)",
  "controllerEndpointAddressSpace": {
    "name": "vpc-90e3f1f4_VPC_SDSN",
    "type": "Vpc"
  },
  "additionalSettings": {
    "total": 4,
    "settings": [
      {
        "name": "SRX_USERNAME",
        "value": "root"
      },
      {
        "name": "SRX_ID_TAG",
        "value": "VSRX"
      },
      {
        "name": "PRIVATE_KEY_FILE",
        "value":
"https://127.0.0.1:8080/api/v2/controller/files/852af83b-0a6d-40c9-b397-4964de59ccc3"
      },
      {
        "name": "SECURITY_GROUP_NAME",
        "value": "aws_sg_block"
      }
    ]
  }
}

```

## Delete a Connector Instance

Use this request to delete a connector instance.

URI	/api/v2/connector/instances/{connectorId}
HTTP method	DELETE
Content-Type	None
Status	<ul style="list-style-type: none"> <li>• 200—Connector is successfully deleted</li> <li>• 500—Unexpected error</li> </ul>

## Get the Connector and its Controller Status

Use this request to check reachability of the connector and the status of its underlying controller.

URI	/api/v2/connector/instances/{connectorId}/heartbeat
HTTP method	GET
Content-Type	None
Status	<ul style="list-style-type: none"> <li>• 200—Connector is reachable and the status of its underlying controller is OK.</li> <li>• 503—Connector is reachable, but the status of its underlying controller is not OK (for example, not reachable or no proper response).</li> <li>• 500—Unexpected error</li> </ul>

## Get All Enforcement Point Devices

Use this request to obtain information on the physical or virtual network devices such as firewall, switches, and routers performing the enforcement-related operations. These operations are either currently managed or to be managed by the associated EMS of Policy Enforcer.

You can perform this request only if the connector supports the ENFORCEMENT\_POINT\_INFO capability.

URI	/api/v2/connector/instances/{connectorId}/enforcement-points
HTTP method	GET

Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns a list of enforcement point devices.</li> <li>• 500—Unexpected error</li> <li>• 501—Connector does not support the ENFORCEMENT_POINT_INFO capability.</li> </ul>

### Usage Example

```

Response:
{
  "device": [
    {
      "authSecretType": "PRIVATE_KEY_FILE",
      "ip": "35.163.47.86",
      "name": "un-ore-vsrx02",
      "privateKeyFile":
      "https://127.0.0.1:8080/api/v2/controller/files/fc6aca5e-f3a7-4d9e-9bac-bf5fbfab4be3",

      "roles": "FIREWALL",
      "tags": [
        {
          "key": "Name",
          "value": "un-ore-vsrx02"
        },
        {
          "key": "vSRX",
          "value": "un-ore-vsrx02"
        }
      ],
      "username": "root"
    }
  ],
  "total": 1
}

```

## Get All Enforcement Subnets

Use this request to obtain information on the subnets containing endpoints that the connector can perform enforcement actions.

This is only available if the connector supports the ENFORCEMENT\_SUBNET\_INFO capability.

URI	/api/v2/connector/instances/{connectorId}/enforcement-subnets
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns a list of enforcement subnets.</li> <li>• 500—Unexpected error</li> <li>• 501—Connector does not support the ENFORCEMENT_POINT_INFO capability.</li> </ul>

### Usage Example

Response:

```
{
  "subnet": [
    "10.0.30.0/24",
    "10.0.10.0/24",
    "10.0.43.0/24",
    "10.0.42.0/24",
    "10.0.41.0/24",
    "10.0.50.0/24",
    "10.0.20.0/24"
  ],
  "total": 7
}
```

## Retrieve Connector Log Files

Use this request to retrieve a zip file containing the log files of a connector server.

URI	/api/v2/connector/logs
-----	------------------------

HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Connector's logs are successfully packaged into a zip file and the zip file is provided in the response.</li> <li>• 204—Logs are not available</li> <li>• 500—Unexpected error</li> </ul>

## Block the Infected Host Threat

Use this request to instruct the connector to perform a remediation action against an infected host threat.

This is available only if the connector supports the `INFECTED_HOST_BLOCK` and `INFECTED_HOST_QUARANTINE` capabilities.

URI	/api/v2/connector/instances/{connectorId}/threats/infected-hosts
HTTP method	POST
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Remediation action is successfully completed</li> <li>• 202—Remediation request is accepted and will be acted upon. However, the action is not yet complete.</li> <li>• 412—Remediation action could not be completed, because details about the specified endpoint are not currently available.</li> <li>• 500—Unexpected error</li> <li>• 501—Connector does not support the requested action</li> </ul>

### Usage Example

```
Request:
{
  "endpoint": {
    "ip": "10.0.30.8",
  },
}
```

```

    "action": "block"
  }

Response:
{
  "action": "block",
  "actionStatus": "SUCCEEDED",
  "endpoint": {
    "additionalInfo": [],
    "ip": "10.0.30.8",
    "macAddress": "00:0C:29:24:8A:F4",
    "nasName": "ex4300-01",
    "nasPort": "ge-0/0/1"
  }
}

```

## Get All Infected Host Threats

Use this request to Obtain the status of all the current infected host threats.

URI	/api/v2/connector/instances/{connectorId}/threats/infected-hosts
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns a list of infected host threats</li> <li>• 500—Unexpected error</li> </ul>

### Usage Example

```

Response:
{
  "total": 1,
  "threats": [

```



```

{
  "action": "block",
  "actionStatus": "SUCCEEDED",
  "endpoint": {
    "additionalInfo": [],
    "ip": "10.0.30.8",
    "macAddress": "00:0C:29:24:8A:F4",
    "nasName": "ex4300-01",
    "nasPort": "ge-0/0/1"
  }
}
]
}

```

## Block the DDoS Attack Threat

Use this request to instruct the connector to perform a remediation action against a Distributed Denial of Service (DDoS) attack threat.

This is available only if the connector supports the DDOS\_BLOCK capability.

URI	/api/v2/connector/instances/{connectorId}/threats/ddos
HTTP method	POST
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Remediation action is successfully completed</li> <li>• 202—Remediation request is accepted and will be acted upon. However, the action is not yet complete.</li> <li>• 500—Unexpected error</li> <li>• 501—Connector does not support the DDOS_BLOCK capability.</li> </ul>

### Usage Example

```
Request:
{
  "sourceIps": [
    "1.1.1.1",
    "2.2.2.2"
  ],
  "enforcementPointIp": "192.168.10.2"
}
```

```
Response:
{
  "sourceIps": [
    "1.1.1.1",
    "2.2.2.2"
  ],
  "enforcementPointIp": "192.168.10.2",
  "actionStatus": "SUCCEEDED"
}
```

## Get a List of DDoS Threats

Use this request to obtain the status of all the current DDoS threats.

URI	/api/v2/connector/instances/{connectorId}/threats/ddos
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>● 200—Returns a list of DDoS threats</li> <li>● 500—Unexpected error</li> </ul>

### Usage Example

```

Response:
{
  "total": 1,
  "threats": [
    {
      "sourceIps": [
        "1.1.1.1",
        "2.2.2.2"
      ],
      "enforcementPointIp": "192.168.10.2",
      "actionStatus": "SUCCEEDED"
    }
  ]
}

```

## Get IP Addresses of the Endpoints

Use this request to query the IP addresses of the endpoints tagged with a specified value.

This is available only if the connector supports the ENDPOINT\_TAGGING\_INFO capability.

URI	/api/v2/connector/instances/{connectorId}/endpoints?tagValue=Web&tagName=Application  <b>NOTE:</b> You can use asterisk (*) for the tag value field to query all endpoints associated with the specified tag name.
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>200—Returns a list of IP addresses of endpoints tagged with specified value.</li> </ul>

### Usage Example

```

Response:
{

```

```

"endpoints": [
  {
    "ip": "10.0.30.64",
    "tag": {
      "key": "Application",
      "value": "Web"
    }
  },
  {
    "ip": "10.0.43.153",
    "tag": {
      "key": "Application",
      "value": "Web"
    }
  },
  {
    "ip": "10.0.41.85",
    "tag": {
      "key": "Application",
      "value": "Web"
    }
  }
],
"total": 3
}

```

## Get All Tag Names and Their Values

Use this request to obtain information on all tag names that are currently referenced by the endpoints of a connector and for each endpoint, its unique set of values.

This is available only if the connector supports the `ENDPOINT_TAGGING_INFO` capability.

URI	/api/v2/connector/instances/{connectorId}/endpoint-tags
HTTP method	GET
Content-Type	application/json

Status

- 200—Returns a list of tag names and their values.
- 500—Unexpected error
- 501—Connector does not support the ENDPOINT\_TAGGING\_INFO capability

## Usage Example

```

Response:
{
  "total": 2,
  "tags": [
    {
      "tagName": "Application",
      "tagValues": [
        "Web",
        "DB"
      ]
    },
    {
      "tagName": "Deployment",
      "tagValues": [
        "Production",
        "Test"
      ]
    }
  ]
}

```

## Register a Listener

Use this request to register a listener to notify any updates from the group of endpoints tagged with a particular value. When changes occur, a POST call is invoked against `postNotificationUrl` using `postNotificationAuthToken`.

URI	/api/v2/connector/instances/{connectorId}/endpoint-tags/group-membership-listeners
HTTP method	POST
Content-Type	application/json

## Status

- 200—Returns a confirmation that the listener is registered successfully.
- 500—Unexpected error
- 501—Connector does not support the ENDPOINT\_TAGGING\_INFO capability

## Usage Example

## Request:

```
{
  "tagName": "Application",
  "tagValue": "Web",
  "postNotificationUrl":
  "https://127.0.0.1:8080/api/v2/connector/endpoint-tag-listener/f3843d51-b80d-4eaf-a381-d86a38c44fc4",

  "postNotificationAuthToken": "4es00z9b18Q0U6WCue9gwKSyeFPpyaZP"
}
```

## Response:

```
{
  "id": "3",
  "uri":
  "http://127.0.0.1:8084/api/v2/connector/instances/2/endpoint-tags/group-membership-listeners/3",

  "tagName": "Application",
  "tagValue": "Web",
  "postNotificationUrl":
  "https://127.0.0.1:8080/api/v2/connector/endpoint-tag-listener/f3843d51-b80d-4eaf-a381-d86a38c44fc4",

  "postNotificationAuthToken": "4es00z9b18Q0U6WCue9gwKSyeFPpyaZP"
}
```

## Get All Registered Listeners

Use this request to obtain the list of all currently registered endpoint tag listeners. Only available if the connector supports the ENDPOINT\_TAGGING\_INFO capability.

URI	/api/v2/connector/instances/{connectorId}/endpoint-tags/group-membership-listeners
HTTP method	GET

Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns a list of registered endpoint tag listeners</li> <li>• 500—Unexpected error</li> <li>• 501—Connector does not support the ENDPOINT_TAGGING_INFO capability</li> </ul>

### Usage Example

```

Response:
{
  "total": 1,
  "listeners": [
    {
      "id": "3",
      "uri":
"http://127.0.0.1:8084/api/v2/connector/instances/2/endpoint-tags/group-membership-listeners/3",

      "tagName": "Application",
      "tagValue": "Web",
      "postNotificationUrl":
"http://127.0.0.1:8080/api/v2/connector/endpoint-tag-listener/f3843d51-b80d-4eaf-a381-d86a38c44fc4",

      "postNotificationAuthToken": "4es00z9b18Q0U6WCue9gwKSYeFPpyaZP"
    }
  ]
}

```

## Get Details of a Listener by ID

Use this request to obtain information about the specified endpoint tag listener. Only available if the connector supports the ENDPOINT\_TAGGING\_INFO capability.

URI	/api/v2/connector/instances/{connectorId}/endpoint-tags/group-membership-listeners/{listenerId}
HTTP method	GET
Content-Type	application/json

Status	<ul style="list-style-type: none"> <li>• 200—Returns information about the endpoint tag listener</li> <li>• 500—Unexpected error</li> </ul>
--------	---

### Usage Example

```

Response:
{
  "id": "3",
  "uri":
"http://127.0.0.1:8084/api/v2/connector/instances/2/endpoint-tags/group-membership-listeners/3",

  "tagName": "Application",
  "tagValue": "Web",
  "postNotificationUrl":
"https://127.0.0.1:8080/api/v2/connector/endpoint-tag-listener/f3843d51-b80d-4eaf-a381-d86a38c44fc4",

  "postNotificationAuthToken": "4es00z9bl8Q0U6WCue9gwKSyeFPpyaZP"
}

```

## Deregister a Listener by ID

Use this request to deregister an endpoint tag listener. This is available only if the connector supports the `ENDPOINT_TAGGING_INFO` capability.

URI	/api/v2/connector/instances/{connectorId}/endpoint-tags/group-membership-listeners/{listenerId}
HTTP method	DELETE
Content-Type	None
Status	<ul style="list-style-type: none"> <li>• 200—Listener is successfully deregistered.</li> <li>• 500—Unexpected error</li> </ul>



## Get a List of Controller Contexts

Use this request to query the specified controller for its list of contexts. This API is supported only for servers that require a controller context value to be specified when creating a connector instance.

URI	/api/v2/connector/controller-contexts
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns a list of controller contexts</li> <li>• 500—Server does not support controller contexts</li> </ul>

### Usage Example

```
GET /api/v2/connector/controller-contexts?controllerClientSecret=
1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc&controllerPort=443&controllerClient
Id=AKIYXUTNS4UIQ5824BMQ&controllerIp=aws.amazon.com
```

Response:

```
[
  {
    "name": "Asia Pacific (Mumbai)",
    "type": "Region"
  },
  {
    "name": "eu-west-3",
    "type": "Region"
  },
  {
    "name": "EU (London)",
    "type": "Region"
  },
  {
    "name": "EU (Ireland)",
    "type": "Region"
  },
  {
    "name": "Asia Pacific (Seoul)",
```

```
    "type": "Region"
  },
  {
    "name": "Asia Pacific (Tokyo)",
    "type": "Region"
  },
  {
    "name": "South America (S\u00e3o Paulo)",
    "type": "Region"
  },
  {
    "name": "Canada (Central)",
    "type": "Region"
  },
  {
    "name": "Asia Pacific (Singapore)",
    "type": "Region"
  },
  {
    "name": "Asia Pacific (Sydney)",
    "type": "Region"
  },
  {
    "name": "EU (Frankfurt)",
    "type": "Region"
  },
  {
    "name": "US East (N. Virginia)",
    "type": "Region"
  },
  {
    "name": "US East (Ohio)",
    "type": "Region"
  },
  {
    "name": "US West (N. California)",
    "type": "Region"
  },
  {
    "name": "US West (Oregon)",
    "type": "Region"
  }
}
```

```

    }
  ]

```

## Get More Information About a Controller

Use this request to query the specified controller for information, such as its set of entities implementing unique address spaces for endpoints.

URI	/api/v2/connector/controller-info
HTTP method	GET
Content-Type	application/json
Status	<ul style="list-style-type: none"> <li>• 200—Returns information about the specified controller.</li> </ul>

### Usage Example

```

GET
/api/v2/connector/controller-info?controllerContext=Asia+Pacific+%28Mumbai%29&
  controllerClientSecret=1ggYYha0OCF0kG9qaJMG0Pgjacd3rStvAeHgHmVc&
  controllerPort=443&controllerClientId=AKIYXUTNS4UIQ5824BMQ&
  controllerIp=aws.amazon.com
Response:
{
  "endpointAddressSpaces": [
    {
      "details": {
        "childAddressSpaces": [
          {
            "name": "ap-south-1a",
            "subnets": [
              "10.0.3.0/24",
              "10.0.254.0/24",
              "10.0.2.0/24",

```

```

        "10.0.5.0/24",
        "10.0.1.0/24"
    ],
    "type": "AvailabilityZone"
}
],
"endpointTags": {
    "total": 1
    "tags": [
        {
            "tagName": "Application",
            "tagValues": [
                "Web",
                "DB"
            ]
        }
    ],
    "subnet": "10.0.0.0/16"
},
"endpointAddressSpace": {
    "name": "vpc-5ff20937_manasahg-vpc",
    "type": "Vpc"
}
},
{
    "details": {
        "childAddressSpaces": [
            {
                "name": "ap-south-1a",
                "subnets": [
                    "10.0.2.0/24",
                    "10.0.1.0/24",
                    "10.0.4.0/24",
                    "10.0.8.0/24",
                    "10.0.254.0/24",
                    "10.0.3.0/24",
                    "10.0.7.0/24",
                    "10.0.6.0/24",
                    "10.0.5.0/24"
                ],
                "type": "AvailabilityZone"
            }
        ]
    }
}

```

```
    ],  
    "endpointTags": {  
      "total": 1  
      {  
        "tagName": "Application",  
        "tagValues": [  
          "Web",  
          "DB"  
        ]  
      }  
    },  
    "subnet": "10.0.0.0/16"  
  },  
  "endpointAddressSpace": {  
    "name": "vpc-b7530fde_abdulh-vpc",  
    "type": "Vpc"  
  }  
}  
]  
}
```

## RELATED DOCUMENTATION

[Understanding the Policy Enforcer Connector API](#) | 37