



RESTful API v4.19 User Guide

1. RESTful API v4.19 User Guide	4
1.1 Overview	8
1.2 Version	11
1.3 Domain (Content) Filters	13
1.4 Domain (Content) Filter Details	18
1.5 Incident Filters	25
1.6 Incident Filter Details	30
1.7 Event Filters	34
1.8 Event Filter Details	38
1.9 Incidents	43
1.10 Incident Types	47
1.11 Incident Details	49
1.12 Events	51
1.13 Event Types	57
1.14 Device Inventory (Asset Management)	59
1.15 Auto Discovery Status	81
1.16 Auto Discovery Profiles	85
1.17 Auto Discovery Profile	91
1.18 Auto Discovery Results	103
1.19 Auto Discovery Result	111
1.20 Product Info	113
1.21 Servers	115
1.22 Servers Details	117
1.23 User Groups	119
1.24 User Group Details	123
1.25 User Group Permissions	125
1.26 Tools	128
1.27 Users	131
1.28 User Details	134
1.29 User's Groups	140
1.30 Global User Settings	142
1.31 Global Password Complexity Settings	145
1.32 View List	148
1.33 View Details	161
1.34 View Objects	166
1.35 Object Attributes	170
1.36 Object Attribute Details	173
1.37 Object Ports	178
1.38 Object Associations	184
1.39 Association Details	187
1.40 Configuration Management	190
1.41 Port Management	196
1.42 Zones	198
1.43 Zone Details	203
1.44 IP SLA Management	209
1.45 IP SLA Creators	218
1.46 IP SLA Pollers	225
1.47 Data Access Interface	228
1.48 Data Access Templates Management – Listing and Creating	232
1.49 Data Access Templates Management – Operations on a Single Template	235
1.50 Data Access – Export Triggering	239
1.51 Flow Data - Listing Devices with Stored Flow Information	243
1.52 Flow Data - Listing Applications Supporting Flow	245
1.53 Flow Data - Time Series History	248
1.54 Maintenance List	253
1.55 Maintenance Details	258
1.56 Meraki Cloud Controllers	264
1.57 Shared Secret Key	266
1.58 Meraki Webhooks	270
1.59 Cisco DNA Center Webhooks	271
1.60 Raise Webhook Events on Cisco DNA Centers	272
1.61 Manage Cisco DNA Center Webhooks	275
1.62 Cisco DNA Centers	276
1.63 Credential Management	277
1.64 Services Hierarchy	289
1.65 Service Detail	295

1.66	OS Services Summary	302
1.67	OS Services Detail	310
1.68	User Defined REST Pollers	314
1.69	User Defined REST Poller Details	335
1.70	Testing User Defined REST Pollers	344
1.71	Custom Webhook Groups	347
1.72	Custom Webhook Group Details	351
1.73	Custom Webhook Rules	355
1.74	Custom Webhook Rule Details	359
1.75	Custom Webhook Endpoints	363
1.76	Custom Webhook Endpoint Details by Group	365
1.77	Custom Webhook Endpoint Details	368
1.78	Custom Webhook Events	370
1.79	Custom Webhook Event Details	372
1.80	Custom Webhook Event Details by Group	374
1.81	List Custom Webhook Payloads	377
1.82	Create Custom Webhook Payloads	379
1.83	License Usage	381
1.84	IPAM - Settings and Scanning	384
1.85	IPAM - Networks	389
1.86	IPAM - Network Details	393
1.87	IPAM - IP Address Pages	399
1.88	IPAM - IP Addresses	401
1.89	IPAM - IP Address Comments and History	403
1.90	IPAM - DHCP Servers	407
1.91	IPAM - DHCP Server Details	410
1.92	Config Sets - List and Create Config Sets	414
1.93	Config Sets - Summary Info	418
1.94	Config Sets - Update and Delete Config Sets	423
1.95	Config Sets - Config Set Details	428
1.96	Config Sets - Config Set Active Config	438
1.97	Config Sets - Populate Config Set from Local Server	439
1.98	Config Sets - Servers	440
1.99	Config Sets - Force Control of Server	443
1.100	Config Sets - Sync	444
1.101	Config Sets - Sync (Change) History	445
1.102	Config Sets - Users	451
1.103	Config Sets - User Details	455
1.104	Config Sets - User Group Membership	457
1.105	Config Sets - List and Add Views	461
1.106	Config Sets - Update, Rename and Delete Views	466
1.107	Config Sets - Content Filters	471
1.108	Config Sets - Incident Filters	475
1.109	Config Sets - Event Filters	478

RESTful API v4.19 User Guide

Version history

Version	Date	Edits
4.0	January 8th, 2019	• Domain Filter Details • Inventory Details • Product Info • Servers • Servers Details • Data Access Interface • Data Access Templates Management – Listing and Creating • Data Access Templates Management – Operations on a Single Template • Data Access – Export Triggering • Flow Data - Listing Devices with Stored Flow Information • Flow Data - Listing Applications Supporting Flow
4.1	March 14th, 2019	• User Details • Global User Settings • Global Password Complexity Settings • Inventory Details
4.2	May 29th, 2019	• Auto Discovery Status • Auto Discovery Profiles • Auto Discovery Profile
4.3	June 27th, 2019	• User Details
4.4	July 22nd, 2019	• Inventory Summary • Servers • Servers Details • Zone Details • Object Attribute Details

4.5	January 17th, 2020	• Overview - OPTIONS Method • Auto Discovery Profiles • Auto Discovery Profile • View List • View Details • Maintenance List • Maintenance Details • Flow Data - Time Series History • Meraki Cloud Controllers • Shared Secret Key • Meraki Webhooks
4.6	August 28th, 2020	• Cisco DNA Center Webhooks • Manage Cisco DNA Center Webhooks • Cisco DNA Centers • Raise Webhook Events on Cisco DNA Centers • View Details • Inventory Summary • Data Access Interface • User's Groups • Version
4.7	October 2nd, 2020	• Inventory Details
4.8	January 7th, 2021	• Inventory Summary • Inventory Details • Auto Discovery Profiles • Auto Discovery Profile
4.9	April 22nd, 2021	• Credential Management • Auto Discovery Status • Auto Discovery Profiles • Auto Discovery Profile • Auto Discovery Results • Auto Discovery Result
4.10	July 23rd, 2021	• Data Access Interface • Data Access - Export Triggering
4.11	October 6th, 2021	• Services Hierarchy • Service Detail • OS Services Summary • OS Services Detail • Events

4.12	December 15th, 2021	• Events
4.13	April 15th, 2022	• User Defined REST Pollers • User Defined REST Poller Details • Testing User Defined REST Pollers • Custom Webhook Groups • Custom Webhook Group Details • Custom Webhook Rules • Custom Webhook Rule Details • Custom Webhook Endpoints • Custom Webhook Endpoint Details by Group • Custom Webhook Events • Custom Webhook Event Details by Group • List Custom Webhook Payloads • Create Custom Webhook Payloads • Events
4.14	July 15th, 2022	• Custom Webhook Group Details • Custom Webhook Endpoint Details • Custom Webhook Event Details • Inventory Summary • Meraki Webhooks
4.15	October 10th, 2022	• Auto Discovery Profile • Auto Discovery Results • User Defined REST Pollers
4.16	November 8th, 2022	• View Objects
4.17	January 16th, 2023	• License Usage

4.18	May 1st, 2023	• Overview • Device Inventory (Asset Management) • Auto Discovery Profile • Custom Webhook Endpoint Details by Group • IPAM Settings and Scanning • IPAM Networks • IPAM Network Details • IPAM IP Address Pages • IPAM IP Addresses • IPAM IP Address Comments and History • IPAM DHCP Servers • IPAM DHCP Server Details • Config Sets - List and Create Config Sets • Config Sets - Summary Info • Config Sets - Update and Delete Config Sets • Config Sets - Config Set Details • Config Sets - Config Set Active Config • Config Sets - Populate Config Set from Local Server • Config Sets - Servers • Config Sets - Force Control of Server • Config Sets - Sync • Config Sets - Sync (Change) History • Config Sets - Users • Config Sets - User Details • Config Sets - User Group Membership • Config Sets - List and Add Views • Config Sets - Update, Rename and Delete Views • Config Sets - Content Filters • Config Sets - Incident Filters • Config Sets - Event Filters
4.19	July 10th, 2023	• Custom Webhook Rules

Overview

- [Introduction](#)
- [OPTIONS Method](#)
- [Multi-Server Resources](#)
- [Versioning](#)
- [Authentication](#)
 - [Access key authentication:](#)

Introduction

This document describes the RESTful API provided by Entuity. The API is accessible via HTTP or HTTPS protocol, dependent on the webserver configuration, and is exposed via URLs under the /api path. For example, you can access the 'info' resource via <https://myserver/api/info> or <https://myserver/api/info> depending on whether or not you have configured Entuity for HTTPS access.

Each resource may support one or more of the following HTTP methods: GET, POST, PUT, DELETE, OPTIONS. Each resource may require input and may produce output. Please see the following documentation for each resource for details on supported methods and input/output details.

Resources can expect input in different forms:

- Query parameters, e.g. <https://myserver/api/someResource?param=value>. Note that 'value' in the example URL must be URL encoded (e.g. 'hello world' must be encoded as 'hello%20world'). For the list of characters that must be encoded, see: https://www.w3schools.com/tags/ref_urlencode.asp.
- HTTP content (Entity). The Entity may be represented as either XML (content-type:application/xml) or JSON (content-type:application/json). Note that when sending entity in the request, you must specify the "content-type" header. If using the curl tool you can use the -H argument to specify the header, e.g. curl -H "content-type:application/json"

Most, but not all, resources will return resource representations in either XML or JSON. You can specify which representation by supplying a 'media' query parameter with a value of 'xml' or 'json', e.g. <https://myserver/api/info?media=xml>. Alternatively, you can specify this using a header field, e.g. Accept: application/xml or Accept: application/json.

Each response has a response code, indicating a success/failure of the request. These are standard HTTP specification response codes:

- 200-299: indicates success.
- 300-399: indicates redirection: clients should repeat request at redirected location.
- 400-499: indicates a problem with a client request.
- 500-599: indicates a problem on a server side.

OPTIONS Method

You can get some simple information on a resource's supported methods by issuing an OPTIONS method against that resource, or you can find all available resources by issuing an OPTIONS method against a root resource. For example, by using curl:

- curl -X OPTIONS <https://myserver/api/>. This will return an ALLOW header containing the supported methods.

Multi-Server Resources

By default, HTTP methods operate on the resources local to the server you are speaking to. However, if the server you are speaking to has remote servers configured, you can work with any of them. You can qualify the server you want to be working with by using a query parameter 'serverId', e.g. <https://mycentralserver/api/info?serverId=long-id-of-the-remote-server>.

Versioning

All resources accessible via `/api/*` can also be accessed via `/api/v2/*`. Clients who wish to remain compatible with future versions of Entuity should use resources under the specific version: `/api/v2/*`. Resources without a version specifier under `/api/*` will contain the latest resource implementations, and may be changed in future releases.

Please note that resources under the specific version `/api/v2/*` may still be changed with new Entuity versions /patches. However, any changes will be limited to compatible changes that are unlikely to break any integrations making use of the API.

Compatible changes include:

- addition of new authentication methods.
- addition of new resources.
- addition of new fields in resource representations returned.
- requiring fewer inputs.
- applying fewer constraints on input.

Incompatible change will be added with a new version number, so the following URLs will be available:

- `/api/v1/*` will use version 1 of the API.
- `/api/v2/*` will use version 2 of the API which will include changes that are not compatible with previous versions.
- `/api/*` Will always track the latest version of the API.

Resources in this document are described using URLs relative to their version base, e.g. resource 'info' can be accessed as `/api/info` or `/api/v2/info`.

Please see the Version section in this document so as to check the latest version of the API.

Authentication

Entuity supports basic HTTP authentication method (RFC 2617). Basic HTTP authentication is widely supported. Please note that basic HTTP authentication is insecure when used without SSL, because passwords are sent as clear text). **Therefore, Entuity recommends using the API over HTTPS.**

If using the curl tool, you can supply '-u username:password' arguments to provide authentication details.

For performance reasons, authentication results are cached on the server for 5 minutes after they are last used.

Note that you can authenticate with any Entuity user and the resources are protected by using the Entuity permission model: Users will only be able to access and modify resources that they have permission to access.

Access key authentication:

Alongside authenticating with a username and password by using the HTTP 'Authorization' header and 'Basic' scheme, you can authenticate with an access key.

The access key can be supplied in the HTTP request's **ENA-API-Key** header. **Note**, if both the 'Authorization' header and the 'ENA-API-Key' are specified, the 'Authorization' header will take precedence and the access key is ignored.

Valid access keys are configured in <ENTUITY_HOME>/etc/accessKeys.properties file. **Note**, this file does not exist by default, and needs to be created by the administrator.

The format of the file is as follows:

- single-line comments begin with a hash # symbol.
- access keys are specified in the following format: <access_key> = <ena_username>, where:
 - <access_key> should be a long string which is hard to guess, and should contain the following characters only:
 - digits (0-9)
 - lower case ASCII characters (a-z)
 - upper case ASCII characters (A-Z)
 - <ena_username> is the name of the existing Entuity account. HTTP requests which authenticate with the provided access key will be performed on behalf of this user.

Example accessKeys.properties:

```
# This is a comment
verystrongsecret = admin
anotherstrongsecret = user
```

For authentication with the access key to be successful, the 'ENA-API-Key' header in the HTTP request must match one of the access keys specified in the file (case sensitive), and the user associated with the access key must already exist in Entuity.

Version

Shows the version of the RESTful API in use.

URL: version

- [Methods Summary](#)
- [GET Method detail](#)
 - [Examples](#)

Methods Summary

- GET Method - shows the version of the RESTful API in use.

GET Method detail

Shows the version of the RESTful API in use.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/version?media=json</code>
OUTPUT	<pre>{ "version" : "v5" }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/version?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <versionInfo version="5" xmlns:ns5="https://www.entuity.com/webrpc" xmlns: ns2="https://www.entuity.com/schemas/webUI" xmlns:ns4="https://www.entuity. com/schemas/eventengine" xmlns:ns3="https://www.entuity.com/schemas/flow"/></pre>

Note, you can specify a particular version of the API function to use in the URL. This feature may help reduce maintenance with third party integrations that use API functions that may change in later releases.

E.g. to specify a particular version of the API to use, place the version number in the URL:

api/{version}/function

INPUT	<code>curl -u admin:admin https://localhost/api/v1/version? media=json</code>
OUTPUT	<pre>{ "version" : "v1" }</pre>

Domain (Content) Filters

Lists the filter id, name and serverId information of available domain (content) filters. Note, from Entuity v21.0 upwards, you can manage content filters within config sets (see 'Config Sets - Content Filters').

URL: domainFilters (example <https://localhost/api/domainFilters?media=json>)

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists available domain filters.
- POST Method - creates new domain filter.

GET Method detail

Lists available domain (content) filters.

Response

Response includes a list of domain filters. Each domain filter has the following attributes:

Name	Description
id	domain filter ID unique to the server.
name	domain filter name.
serverId	Entuity Server ID on which resource resides.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/domainFilters?media=json</code>
--------------	---

OUTPUT	<pre>{ "items" : [{ "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : "1", "name" : "All Objects" }, { "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : "2", "name" : "Infrastructure Only" }], "count" : 2 }</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/domainFilters?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="All Objects" id="1" serverId=" 821c3e87-bcdf-4fef-a5e8-7d2524928d96" /> <item xsi:type="namedItem" name="Infrastructure Only" serverId=" 821c3e87-bcdf-4fef-a5e8-7d2524928d96" /> </items></pre>

POST Method details

Creates a new domain filter.

Request

Name	Description
name	filter name.
rules	the array of rules defining a filter.

Response

Name	Description
------	-------------

name	filter name.
systemFilter	whether the filter is a system filter.
rules	The array of rules defining a filter.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/domainFilters?media=json -X POST -H "content-type:application/json" -d \ ' { "name" : "A", "rules" : [{ "SRCTYPE" : "4", "DEVNAME" : "Two", "ZONENAME" : "None" }] }'</pre>
OUTPUT	<pre>{ "name" : "A", "systemFilter" : false, "rules" : [{ "SRCTYPE" : "4", "DEVNAME" : "Two", "ZONENAME" : "None" }] }</pre>

INPUT	<pre> curl -u admin:admin https://localhost/api/domainFilters?media=xml -X POST -H "content-type:application/xml" -d \ '<domainFilterInfo systemFilter="false" name="B"> <rules> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">4< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Two< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">ZONENAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">None< /value> </entries> </rules> </domainFilterInfo>' </pre>
-------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <domainFilterInfo systemFilter="false" name="B"> <rules> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Two< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">4< /value> </entries> </rules> </domainFilterInfo></pre>
---------------	--

Domain (Content) Filter Details

Represents a set of operations on a particular domain filter. Note, from Entuity v21.0 upwards, you can manage content filters within config sets (see 'Config Sets - Content Filters').

URL: domainFilters/{filterId}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - shows detailed information about the filter.
- PUT Method - modifies the parameters of the filter.
- DELETE Method - deletes the filter.

GET Method details

Shows detailed information about the selected domain (content) filter.

Response

General parameters:

Name	Description
name	domain filter name.
systemFilter	whether the filter is a system filter.
rules	array of rules defining a filter, as per Rule definition section below.

Rule definitions - a single rule may contain at most one of the following parameters, the exception being that SRCTYPE has to be present in the rule definition:

Name	Description
SRCTYPE	source type, one of the following (case-insensitive): • PORT • DEVICE • VLAN • APPLICATION • SERVICE
DEVTYPE	device type to which this filter refers.
DEVNAME	name of an object (port, device, application etc).
ZONENAME	name of a zone.
IPLE	IP low-end, used for defining IP ranges.
IPHE	IP high-end, used for defining IP ranges.
MANAGEMENT_ONLY	Management IP Only.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/domainFilters/1002?media=json</code>
OUTPUT	<pre>{ "name" : "A", "systemFilter" : false, "rules" : [{ "SRCTYPE" : "DEVICE", "DEVNAME" : "Two", "ZONENAME" : "All" }] }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/domainFilters/1002?media=xml</code>
--------------	---

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <domainFilterInfo systemFilter="false" name="A"> <rules> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">ZONENAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">All</value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Two</value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">DEVICE</value> </entries> </rules> </domainFilterInfo> </pre>
---------------	---

PUT Method details

Modifies the selected domain filter.

Request

Name	Description
name	Filter name
rules	The array of rules defining a filter, as per Rule Definition section above.

Response

The modified filter, as the detailed GET method would return it (see Rule Definition section above).

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/domainFilters/1002?media=json -X PUT -H "content-type:application/json" -d \ { "name" : "B", "rules" : [{ "SRCTYPE" : "APPLICATION", "DEVTYPE" : "158", "ZONENAME" : "None" }] }</pre>
OUTPUT	<pre>{ "name" : "B", "rules" : [{ "SRCTYPE" : "APPLICATION", "DEVTYPE" : "158", "ZONENAME" : "None" }] }</pre>

INPUT	<pre> curl -u admin:admin https://localhost/api/domainFilters/1002?media=xml - X PUT -H "content-type:application/xml" -d \ '<domainFilterInfo systemFilter="false" name="B"> <rules> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" >APPLICATION</value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">ZONENAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">None< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Two< /value> </entries> </rules> </domainFilterInfo>' </pre>
-------	--

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <domainFilterInfo systemFilter="false" name="B"> <rules> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance">ZONENAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">None< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Two< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" >APPLICATION</value> </entries> </rules> </domainFilterInfo> </pre>
---------------	--

DELETE Method details

Deletes the selected domain filter.

Request

No additional parameters needed.

Response

Gives the message "OK" if operation was successful. Otherwise, gives an error description.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/domainFilters/1002?media=json -X DELETE</code>
OUTPUT	<pre>{ "message" : "OK" }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/domainFilters/1002?media=xml -X DELETE</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>

Incident Filters

List available incident filters, and create new filters. Note, from Entuity v21.0 upwards, you can manage incident filters within config sets (see 'Config Sets - Incident Filters').

URL: incidentFilters

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists available incident filters.
- POST Method - creates new incident filter.

GET Method detail

Lists available incident filters.

Response

Response includes a list of incident filters. Each incident filter has the following attributes:

Name	Description
serverId	Entuity Server Id on which resource resides.
id	incident filter id unique to the server.
name	incident filter name.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/incidentFilters?media=json</code>
--------------	---

OUTPUT	<pre>{ "items" : [{ "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : "1", "name" : "All Incidents" }, { "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : "2", "name" : "TestIncidents" }], "count" : 2 }</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/incidentFilters?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="All Incidents" id="1" serverId=" 821c3e87-bcdf-4fef-a5e8-7d2524928d96" /> <item xsi:type="namedItem" name="TestIncidents" id="2" serverId=" 821c3e87-bcdf-4fef-a5e8-7d2524928d96" /> </items></pre>

POST Method details

Creates new incident filter.

Request

Name	Description
name	filter name.
selectedNames	array of filter names.
passIP	whether the filter should include devices that are not under management.

Response

Name	Description
name	filter name.
selectedNames	array of filter names.
systemFilter	whether the filter is a system filter.
passIP	whether the filter should include devices that are not under management.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/incidentFilters?media=json -X POST -H "content-type:application/json" -d \ ' { "name" : "A", "selectedNames" : ["AP Antenna Host Count Abnormality", "AP Not Associated With Controller"], "passIP" : false }'</pre>
OUTPUT	<pre>{ "name" : "A", "selectedNames" : ["AP Antenna Host Count Abnormality", "AP Not Associated With Controller"], "systemFilter" : false, "passIP" : false }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/incidentFilters?media=xml -X POST -H "content-type:application/xml" -d \ '<eventFilterInfo systemFilter="false" passIP="true" name="A"> <selectedNames> <filterName>AvailMonitor High Latency</filterName> <filterName>AvailMonitor Application Problem</filterName> </selectedNames> </eventFilterInfo>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="true" name="A"> <selectedNames> <filterName>AvailMonitor High Latency</filterName> <filterName>AvailMonitor Application Problem</filterName> </selectedNames> </eventFilterInfo></pre>

Incident Filter Details

Show details about a particular incident filter, modify the parameters of a filter, and delete a filter. Note, from Entuity v21.0 upwards, you can manage incident filters within config sets (see 'Config Sets - Incident Filters').

URL: incidentFilters/{filterId}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - shows detailed information about the filter.
- PUT Method - modifies the parameters of the filter.
- DELETE Method - deletes the filter.

GET Method details

Shows detailed information about the selected incident filter.

Response

Name	Description
name	incident filter name.
systemFilter	whether this filter is a system filter.
rules	list of rules in this filter.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/incidentFilters/2?media=json</code>
-------	---

OUTPUT	<pre>{ "name" : "A", "selectedNames" : ["AP Antenna Host Count Abnormality", "AP Not Associated With Controller"], "systemFilter" : false, "passIP" : false }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/domainFilters/2?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="true" name="A"> <selectedNames> <filterName>AvailMonitor High Latency</filterName> <filterName>AvailMonitor Application Problem</filterName> </selectedNames> </eventFilterInfo></pre>

PUT Method details

Modifies the selected incident filter.

Request

Name	Description
name	filter name.
rules	list of rules defining the filter.

Response

Name	Description
name	filter name.
systemFilter	whether the filter is system one.

rules	list of rules defining a filter.
--------------	----------------------------------

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/incidentFilters/2?media=json -X PUT -H "content-type:application/json" -d \ '{ "name" : "B", "selectedNames" : ["AP Antenna Host Count Abnormality", "AP Antenna Power Change Frequency High", "AP Not Associated With Controller"], "systemFilter" : false, "passIP" : true }'</pre>
OUTPUT	<pre>{ "name" : "B", "selectedNames" : ["AP Antenna Host Count Abnormality", "AP Antenna Power Change Frequency High", "AP Not Associated With Controller"], "systemFilter" : false, "passIP" : true }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/incidentFilters/2?media=xml-X PUT -H "content-type:application/xml" -d \ '<eventFilterInfo systemFilter="false" passIP="false" name="B"> <selectedNames> <filterName>AP Antenna Host Count Abnormality</filterName> <filterName>AP Antenna Power Change Frequency High</filterName> <filterName>AP Not Associated With Controller</filterName> </selectedNames> </eventFilterInfo>'</pre>
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="false" name="B"> <selectedNames> <filterName>AP Antenna Host Count Abnormality</filterName> <filterName>AP Antenna Power Change Frequency High</filterName> <filterName>AP Not Associated With Controller</filterName> </selectedNames> </eventFilterInfo></pre>
---------------	---

DELETE Method details

Deletes the selected incident filter.

Request

No additional parameters needed.

Response

"OK" message if operation was successful, and an error description otherwise.

Examples

INPUT	curl -u admin:admin https://localhost/api/incidentFilters/2?media=json -X DELETE "content-type:application/json"
OUTPUT	<pre>{ "message" : "OK" }</pre>

INPUT	curl -u admin:admin https://localhost/api/incidentFilters/2?media=xml -X DELETE
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>

Event Filters

Lists minimal information about available event filters. Note, from Entuity v21.0 upwards, you can manage event filters within config sets (see 'Config Sets - Event Filters').

URL: eventFilters

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists available event filters.
- POST Method - creates a new event filter.

GET Method detail

Lists available event filters.

Response

Response includes a list of event filters. Each event filter has the following attributes:

Name	Description
id	Event filter id unique to the server
name	Event filter name
serverId	Entuity Server Id on which resource resides

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/eventFilters?media=json</code>
--------------	--

OUTPUT	<pre>{ "items": [{ "serverId": "98eb1254-0d30-4c7d-8910-ac610cd5e351", "id": "1001", "name": "A" }, { "serverId": "98eb1254-0d30-4c7d-8910-ac610cd5e351", "id": "1", "name": "All Events" }], "count": 2 }</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/eventFilters?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="A" id="1001" serverId="98eb1254-0d30-4c7d-8910-ac610cd5e351" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="All Events" id="1" serverId="98eb1254-0d30-4c7d-8910-ac610cd5e351" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>

POST Method details

Creates a new event filter.

Request

Event filter parameters

Name	Description
name	Filter name

selectedNames	The array of filter names
passIP	Whether the filter should include devices not under management

Response

The newly created entry, as detailed GET method would return it.

Name	Description
name	Filter name
selectedNames	The array of filter names
systemFilter	Whether the filter is a system filter
passIP	Whether the filter should include devices not under management

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/eventFilters?media=json -X POST -H "content-type:application/json" -d \ ' { "name" : "A", "selectedNames" : ["AvailMonitor Application Unavailable", "AvailMonitor High Latency Reaching Application Cleared"], "passIP" : true }'</pre>
OUTPUT	<pre>{ "name" : "A", "selectedNames" : ["AvailMonitor Application Unavailable", "AvailMonitor High Latency Reaching Application Cleared"], "systemFilter" : false, "passIP" : true }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/eventFilters?media=xml -X POST -H "content-type:application/xml" -d \ '<eventFilterInfo systemFilter="false" passIP="true" name="A"> <selectedNames> <filterName>AvailMonitor Application Unavailable</filterName> <filterName>AvailMonitor High Latency Reaching Application Cleared< /filterName> </selectedNames> </eventFilterInfo>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="true" name="A"> <selectedNames> <filterName>AvailMonitor Application Unavailable</filterName> <filterName>AvailMonitor High Latency Reaching Application Cleared< /filterName> </selectedNames> </eventFilterInfo></pre>

Event Filter Details

Represents a set of operations on a particular event filter. Note, from Entuity v21.0 upwards, you can manage event filters within config sets (see 'Config Sets - Event Filters').

URL: eventFilters/{filterId}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - shows detailed information about the event filter.
- PUT Method - modifies the parameter of the event filter.
- DELETE Method - deletes the event filter.

GET Method details

Shows detailed information about the selected event filter.

Response

Name	Description
name	filter name.
selectedNames	array of filter names.
systemFilter	whether the filter is a system filter.
passIP	whether the filter should include devices not under management.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/eventFilters/1001?media=json</code>
OUTPUT	<pre>{ "name": "A", "selectedNames": ["AvailMonitor Application Unavailable", "AvailMonitor High Latency Reaching Application Cleared"], "systemFilter": false, "passIP": true }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/eventFilters/1001?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="true" name="A"> <selectedNames> <filterName>AvailMonitor Application Unavailable</filterName> <filterName>AvailMonitor High Latency Reaching Application Cleared< /filterName> </selectedNames> </eventFilterInfo></pre>

PUT Method details

Modifies the parameters of the selected event filter.

Request

Name	Description
name	filter name.
selectedNames	array of filter names.
passIP	whether the filter should include devices not under management.

Response

Name	Description
name	filter name.
selectedNames	array of filter names.
systemFilter	whether the filter is a system filter.
passIP	whether this filter should include devices not under management.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/eventFilters/1001?media=json -X PUT -H "content-type:application/json" -d \ '{ "name" : "B", "selectedNames" : ["AvailMonitor Application Unavailable", "WAN Port Low Outbound Utilization Cleared", "AvailMonitor High Latency Reaching Application Cleared"], "passIP" : false }'</pre>
OUTPUT	<pre>{ "name" : "B", "selectedNames" : ["WAN Port Low Outbound Utilization Cleared", "AvailMonitor Application Unavailable", "AvailMonitor High Latency Reaching Application Cleared"], "systemFilter" : false, "passIP" : false }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/eventFilters/1001?media=xml -X PUT -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="false" name="B"> <selectedNames> <filterName>WAN Port Low Outbound Utilization Cleared</filterName> <filterName>AvailMonitor Application Unavailable</filterName> <filterName>AvailMonitor High Latency Reaching Application Cleared< /filterName> </selectedNames> </eventFilterInfo>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventFilterInfo systemFilter="false" passIP="false" name="B"> <selectedNames> <filterName>WAN Port Low Outbound Utilization Cleared</filterName> <filterName>AvailMonitor Application Unavailable</filterName> <filterName>AvailMonitor High Latency Reaching Application Cleared< /filterName> </selectedNames> </eventFilterInfo></pre>

DELETE Method details

Deletes the selected event filter.

Request

No additional parameters needed.

Response

Gives the message "OK" if operation was successful. Otherwise, gives an error description.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/eventFilters/1001?media=json - X DELETE</pre>
--------------	--

OUTPUT	<pre>{ "message" : "OK" }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin https://localhost/api/eventFilters/1001?media=xml -X DELETE</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>

Incidents

Lists available incidents.

URL: incidents

- [Methods Summary](#)
- [GET Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists available incidents.

GET Method details

Lists available incidents.

Request

Name	Description
updateId	incident update identifier, use the updateId from the last result set to get new incidents.
view	incidents filtered by view.
mask	incident severity mask, severities can be added together to request multiple severities together 1 = Info, 2 = Minor, 4 = Major, 8 = Server, 16 = Critical e.g. mask=24 would return both Server and Critical incidents.
states	incident state, either "open", "closed", "expire" and "all". Multiple incident states can be specified together e.g. states=open&states=closed
openedFrom	incident opening time lower bound.
openedTo	incident opening time upper bound.
closedFrom	incident closing time lower bound.
closedTo	incident closing time upper bound.

Response

Response includes an array of event. Each event has following attributes:

Name	Description
description	event description.
details	event details.
objectKeyInfo	object key, consisting of : swld StormWorks Identifier (integer) compld Classic component identifier (4 integers)
severity	incident severity 2 = Information, 4 = Minor, 6 = Major, 8 = Severe, 10 = Critical
sourceDescription	source description.
impactDescription	impact description.
timeStamp	event timestamp.
id	incident Identifier.
state	incident state.
annotation	incident annotation.
eventCount	number of events contributing to the incident.
extraAttribs	attribute key value pairs for the incident.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/incidents?media=json</code>
--------------	---

OUTPUT	<pre> { "incidents" : [{ "description" : "Entuity Server Component Problem", "details" : "Reason for event", "objectKeyInfo" : { "swId" : 815, "compId" : { "ids" : [1, 2, 3, 0] } }, "severity" : 6, "sourceDescription" : "London01", "impactDescription" : "", "timeStamp" : 1540900781, "id" : 69, "state" : "open", "annotation" : "test annotation", "eventCount" : 1, "extraAttribs" : { "attribKey" : "Attrib value" } }, { ...removed for brevity... }], "updateId" : 799, "count" : 2 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/incidents?media=xml
--------------	--

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<incidentsCollection count="2" updateId="799">
  <incidents>
    <incident id="69" eventCount="1" state="open" annotation=""
severity="6" timeStamp="1540900781" id="69">
      <description>Entuity Server Component Problem</description>
      <details>Reason for event</details>
      <objectKeyInfo>
        <swId>815</swId>
        <compId>
          <ids>
            <ids>1</ids>
            <ids>2</ids>
            <ids>3</ids>
            <ids>0</ids>
          </ids>
        </compId>
      </objectKeyInfo>
      <sourceDescription>Huge</sourceDescription>
      <impactDescription></impactDescription>
      <extraAttribs>
        <entry>
          <key>attribKey</key>
          <value>Attrib value</value>
        </entry>
      </extraAttribs>
    </incident>
    <incident
...removed for brevity...
    </incident>
  </incidents>
</incidentsCollection>
```

Incident Types

Lists incident types.

URL: incidentTypes

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists available incident types.

GET Method details

Lists available incident types.

Response

Response includes an array of event types, each of them having the following format:

Name	Description
name	name of the incident type.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/incidentTypes?media=json</code>
OUTPUT	<pre>[{ "name" : "AP Antenna Channel Change Frequency High" }, { ...removed for brevity... }, { "name" : "Wireless Controller High Number of Connected APs" }]</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/incidentTypes?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="256"> <item xsi:type="incidentType" name="AP Antenna Channel Change Frequency High" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> ...removed for brevity... <item xsi:type="incidentType" name="Wireless Controller High Number of Connected APs" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>

Incident Details

Change state, annotation and attributes of an incident.

URL: incidents/{incidentID}

- [Methods Summary](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- PUT Method - change incident state, annotation, attributes.

PUT Method details

Change incident state, annotation and attributes.

Request

All the fields in an incident request are optional so each can be updated independently.

Name	Description
state	change state. Valid states are "open", "closed" and "expired". Note: It is not possible to change the state of an incident if it has already expired.
annotation	change the annotation for an incident.
attribute	update an attribute for an incident. A key and value must be provided (see examples below). Note: The attribute must have already been defined in the active event project.

Response

Name	Description
message	confirmation message.

Examples

INPUT	<pre>curl -X PUT -u admin:admin https://localhost/api/incidents/69? media=json -H "content-type:application/json" -d \ '{ "state" : "open", "annotation" : "New annotation", "attribute" : {"key" : "attribKey", "value": "New value" } }'</pre>
OUTPUT	<pre>{ "message" : "Incident opened, Attribute 'attribKey' updated" }</pre>

INPUT	<pre>curl -X PUT -u admin:admin https://localhost/api/incidents/69?media=xml - H "content-type:application/xml" -d \ '<incidentInfo> <state>open</state> <annotation>New annotation</annotation> <attribute key="attrib1" value="New value"></attribute> </incidentInfo>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Incident opened, Attribute 'attribKey' updated</message> </statusInfo></pre>

Events

List available events, or raise a new event.

URL: events

- [Methods Summary](#)
- [GET Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Request](#)
 - [Examples](#)

Methods Summary

- GET Method - lists available events.
- POST Method - raises an event.

GET Method details

Lists available events.

Request

Name	Description
updateId	event update identifier, use the updateId in the last result set to get new events.
openedFrom	event opening time lower bound.
openedTo	event opening time upper bound.
closedFrom	event closing time lower bound.
closedTo	event closing time upper bound.
view	events filtered by view.
mask	event severity mask, severities can be added together to request multiple severities together. 1 = Info, 2 = Minor, 4 = Major, 8 = Sever, 16 = Critical e.g. mask=24 would return both Sever and Critical events.

States	event state, either “open”, “closed”, “finalized” and “all”. Multiple event states can be specified together. e.g. states=open&states=closed
---------------	---

Response

Response includes an array of events. Each event has the following attributes:

Name	Description
description	event description.
details	event details.
objectKeyInfo	object key, consisting of : swld StormWorks Identifier (integer) compld Classic component identifier (4 integers)
severity	event severity where : 2 = Information, 4 = minor, 6 = major, 8 = severe, 10 = critical
description	event Description.
sourceDescription	source description.
impactDescription	impact description.
timeStamp	event timestamp.
id	event identifier.
eventNumber	event number.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/events?media=json</code>
--------------	--

OUTPUT	<pre>{ "events" : [{ "description" : "Entuity Server Started", "details" : "Restarted", "objectKeyInfo" : { "swId" : 0, "compId" : { "ids" : [4096, 0, 0, 0] } }, "severity" : 2, "sourceDescription" : "Entuity server London01", "impactDescription" : "Entuity service", "timeStamp" : 1540894535, "id" : 786437, "eventNumber" : 480 }, { ...removed for brevity... }], "updateId" : 479, "count" : 459}]</pre>
---------------	--

INPUT	<pre>curl -u admin:admin https://localhost/api/events?media=xml</pre>
--------------	---

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <eventsCollection count="459" updateId="479"> <events> <event eventNumber="480" severity="2" timeStamp="1540894535" id=" 786437"> <description>Entuity Server Started</description> <details>Restarted</details> <objectKeyInfo> <swId>0</swId> <compId> <ids> <ids>4096</ids> <ids>0</ids> <ids>0</ids> <ids>0</ids> </ids> </compId> </objectKeyInfo> <sourceDescription>Entuity server London01</sourceDescription> <impactDescription>Entuity service</impactDescription> </event> <event> ...removed for brevity... </event> </events> </eventsCollection> </pre>
---------------	--

POST Method details

Raises an event.

Request

Name	Description
id	Event type
source	The name to display for the event source field. If not specified, the object key's name is displayed (see below).

objectKeyInfo	Object key, consisting of: swId : StormWorks Identifier (integer) compId : Classic component identifier (4 integers)
name	Event name
severity	Event Severity 2 = Information, 4 = minor, 6 = major, 8 = severe, 10 = critical
reason	Reason for event
ImpactDescription	Impact description
externalId	extra identifier that can be included to the event, e.g. a custom event attribute. This is a string field.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/events?media=json -X POST -H "content-type:application/json" -d \ '{ "reason" : "API test", "name" : "New Event", "source" : "Event source API test", "id" : 786441, "impactDescription" : "Entuity service", "severity" : 6, "objectKeyInfo" : { "swId" : 12345, "compId" : { "ids" : [4096, 0, 0, 0] } }, "externalId" : "12" }'</pre>
OUTPUT	<pre>{ "message" : "Event sent" }</pre>

INPUT	<pre> curl -u admin:admin https://localhost/api/events -X POST -H "content-type:application/xml" -d \ '<eventInfo> <reason>API test</reason> <name>New event</name> <details>New event details</details> <id>786441</id> <impactDescription>Entuity service</impactDescription> <severity>2</severity> <objectKeyInfo> <swId>0</swId> <compId> <ids> <ids>4096</ids> <ids>0</ids> <ids>0</ids> <ids>0</ids> </ids> </compId> </objectKeyInfo> </eventInfo>' </pre>
OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Event sent</message> </statusInfo> </pre>

Event Types

List available event types.

URL: eventTypes

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists available event types.

GET Method details

Lists available event types.

Response

Response includes an array of event types, each of them having the following format:

Name	Description
name	name of the event type.
severity	severity of the event type.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/eventTypes?media=json</code>
-------	--

OUTPUT	<pre>[{ "name" : "ATM VCC High Inbound Utilization", "severity" : 6 }, { ...removed for brevity... }, { "name" : "WAN Port Low Outbound Utilization Cleared", "severity" : 2 }]</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/eventTypes?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="322"> <item xsi:type="eventType" severity="6" name="ATM VCC High Inbound Utilization" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> ...removed for brevity... <item xsi:type="eventType" severity="2" name="WAN Port Low Outbound Utilization Cleared" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>

Device Inventory (Asset Management)

List the inventory devices (assets) on a server, and add a new device to the inventory.

URL: inventory

List details of, update or delete an inventory device (asset).

URL: inventory/{deviceId}

- [Method Summary](#)
- [Note on the use of credential sets](#)
- [GET Method detail - inventory](#)
 - [Response data keys](#)
 - [Examples](#)
- [POST Method detail - inventory](#)
 - [URL parameters](#)
 - [Request Parameters](#)
 - [Response](#)
 - [Examples](#)
- [GET Method detail - inventory/{deviceId}](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Examples](#)
- [PUT Method detail - inventory/{deviceId}](#)
 - [Request Parameters](#)
 - [Response data](#)
 - [Examples](#)
- [DELETE Method details - inventory/{deviceId}](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method Summary

inventory:

- GET Method - list the contents of the inventory (XML, JSON).
- POST Method - queue a device to add to the inventory of a server (XML, JSON).

inventory/{deviceId}:

- GET Method - list details of an inventory device (XML, JSON).
- PUT Method - change an inventory device's details (XML, JSON).
- DELETE Method - remove a device from the inventory.

Note on the use of credential sets

From **Entuity v19.0 upwards**, users can create shared credential sets that can then be bulk-applied to multiple assets. Entuity recommends that you add devices using these shared credential sets. Please see below for the attributes required to define different types of credentials (credentials can be specified or seen by the corresponding credential name):

SNMP and virtualization-type devices:

- credentialId

Config Management:

- cmCredentialId
- cmCredentialName

Server or storage hardware:

- hwConnMethod
- hwConnPort
- hwCredentialId

Server OS:

- osConnMethod
- osConnPort
- osCredentialId

User Defined REST Poller:

- udConnMethod
- udConnPort

GET Method detail - inventory

List the contents of the inventory, in either XML or JSON formats.

Response data keys

Name	Description
Count	number of devices in the inventory.
Items	list of device summary details: • name: Display Name • Id: Unique Identifier per server • polledName: DNS name or IP address • serverId: server identifier

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/inventory?media=json</code>
--------------	---

OUTPUT	<pre>{ "items": [{ "serverId": "34564e92-3b4f-43ba-94a8-04309c0e48fe", "id": "2", "name": "A", "added": true, "polledName": "10.66.24.2" }], "count": 1 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/inventory?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="device" added="true" polledName="10.66.24.2" name="A" id="2" serverId="34564e92-3b4f-43ba-94a8-04309c0e48fe" xmlns:xsi=" https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>

POST Method detail - inventory

Queue a device to add to the inventory of a server, via either XML or JSON formats.

URL parameters

Note: URL parameters are appended to the end of the URL and are delimited from the main part of the URL using a question mark (?) character, see the example below.

Name	Description
allowduplicateIps	allow duplicate IP addresses. By default, ENA will not manage a device that has IP addresses that have already been seen on an existing managed device. The duplicateIps flag can be used to override this behavior. The flag will accept the values; "false", "no", "0", "true", "yes" or "1". To override ENA's default behavior, set this flag to true. For default behaviour set the flag to false or omit the flag altogether.

Request Parameters

Name	Description
cmCredentialId	ID of the config management credential set.
cmCredentialName	name of the config management credential - this name will only be displayed if it is a shared credential (i.e. not a local credential).
configurationManagementLicense	if Configuration Management is enabled ('true') or disabled ('false'), by default false.
credentialId	ID of the shared credential set, used only for SNMP and virtualization assets. The credential must be a valid credential for the specified device type. credentialId is based on the credentials you have created and therefore can be any value. E.g. if credentialId = 1 is an SNMP credential, and the device requires an API Key credential, the request will then fail.
deviceType	Device type, one of: • Hub. • Token Ring Switch. • Ethernet Switch. • ATM Switch. • Router. • Blade Center. • User Created Node. • VM Platform. • Autonomous WAP. • Firewall. • VPN. • Managed Host. • Non-SNMP Device. • Unclassified (Full). • PoE Midspan Injector. • Load Balancer. • SSL Proxy. • Unclassified. • Wireless Controller. • Uninterruptible Power Supply. • Matrix Switch. • Wide Area Application Service. • Multiplexer. • SDN Controller. • Cloud Controller. • Custom Device.

hwConnMethod	connection method used to the hardware asset, which is the collector name based on the device type being added, from the following: Server: • dellidracssh • hpilossh • idracsnmp • hpinsightsnmp • ciscoucscimcsnmp • openmanagesnmp Storage: • naviseccli • dellisilonssh • nimble • vnxcontrolssh • dellisilonrest • uemcli • netappsnap • datadomainsnmp • dellequallogicsnmp • datadomain
hwConnPort	connection port used to the hardware asset.
hwCredentialId	ID of the hardware asset credential set.
managementLevel	management level: • FULL. • FULL_NO_PORTS. • BASIC. • PING_ONLY. • WEB.
name	device name.
nameUsing	device name is determined using one of the following: CUSTOMNAME. IPADDRESS. POLLEDNAME. RESOLVABLENAME. RESOLVABLENAMEFQ. SYSTEMNAME.

osConnMethod	connection method used to the OS, which is the collector name based on the device type being added, from the following: • osaixssh • ossolarisssh • oslinuxssh • oswinrm
osConnPort	connection port used to the OS.
osCredentialId	ID of the OS credential set.
polledName	DNS name or IP Address.
protocol	transport protocol: IPv4, IPv6.
snmpBulk	whether SNMPBulkGet is used ('true' or 'false', by default false).
snmpPDUSize	maximum size of SNMP PDU. 0 = system default
snmpRetry	number of SNMP retries. 0 = system default
snmpTimeout	SNMP timeout in seconds. 0 = system default
snmpType	SNMP type: • v1. • v2c. • v3. • v1/2.
snmpPort	SNMP port number (if omitted, the default value will be used).
udConnMethod	connection method used to the user defined object.
udConnPort	connection port used to the user defined object.
webURL	web URL, applicable to the following asset types: • Oracle VM Manager • VMWare vCenter or ESXi • Cisco APIC • Cisco DNA Center • Meraki SD Controller • vManage SD Controller

Response

200 - OK (Device is queued to be added to the inventory.)

The status info message will say "Queued" or "Replaced", depending on whether this was the first attempt to add a device, or if there was a previous failed attempt to add the device with the same **polledName**.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ '{"polledName" : "10.66.23.1", "managementLevel" : "FULL", "protocol" : "IPv4", "snmpType" : "v2c", "credentialId" : "1"}'</pre>
OUTPUT	<pre>{ "message": "Queued" }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=xml -X POST -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <inventoryDevice polledName="10.66.23.1" managementLevel="FULL" protocol="IPv4" snmpType="v2c" credentialId="1" />'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Queued</message> </statusInfo></pre>

To add a device that is not a VM Platform, Cloud Controller, SDN Controller, Ping Only, or Custom Device:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST - H "content-type:application/json" -d \ ' { "nameUsing" : "POLLEDNAME", "polledName" : "10.66.33.2", "managementLevel" : "FULL", "protocol" : "IPv4", "snmpType" : "v2c", "credentialId" : "1", } '</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

To specify a Ping Only device:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST - H "content-type:application/json" -d \ ' { "nameUsing" : "POLLEDNAME", "polledName" : "10.66.33.2", "managementLevel" : "PING_ONLY", "protocol" : "IPv4" } '</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

To specify a device type such as VM Platform, Cloud Controller, SDN Controller:

i). To add an Amazon Web Services platform:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ '{"managementLevel" : "WEB", "polledName" : "AWS", "deviceType" : "VM Platform", "credentialId" : "4"}'</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

ii). To add a Meraki Cloud Controller platform:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ '{"managementLevel" : "WEB", "polledName" : "Meraki", "deviceType" : "Cloud Controller", "webURL" : "https://api.meraki.com/api/v0/", "credentialId" : "3"}'</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

iii). To add an Azure platform:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ ' { "managementLevel" : "WEB", "polledName" : "AZURE", "deviceType" : "VM Platform", "credentialId" : "5" }'</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

iv). To add a Cisco APIC SDN Controller:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ ' { "managementLevel" : "WEB", "polledName" : "APIC", "deviceType" : "SDN Controller", "webURL" : "https://myapicname.cisco.com/api/", "credentialId" : "6" }'</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

v). To add a VMWare Platform:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ '{"managementLevel" : "WEB", "polledName" : "vortex", "deviceType" : "VM Platform", "webURL" : "https://vortex/sdk", "credentialId" : "6"}'</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

vi). To add a Viptela device:

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory?media=json -X POST -H "content-type:application/json" -d \ '{"managementLevel" : "WEB", "polledName" : "Viptela1", "deviceType" : "Cloud Controller", "webURL" : "https://api.viptela.com/api/v1", "credentialId" : "6"}'</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

vii). To add a Cisco DNA Center:

INPUT	<pre>curl -u admin:admin https://localhost/api /inventory?media=json -X POST -H "content-type: application/json" -d \ ' { "managementLevel" : "WEB", "polledName" : "DNAC1", "deviceType" : "Cisco DNA Center", "webURL" : "https://1.2.3.4", "credentialId" : "6" } '</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

viii). To add an Cisco UCS server device:

INPUT	<pre>curl -u admin:admin https://localhost/api /inventory?media=json -X POST -H "content-type: application/json" -d \ ' { "managementLevel" : "CLI", "polledName" : "CISCO UCS", "deviceType" : "Server", "hwConnMethod" : "ciscoucscimcsnmp", "hwConnPort" : "0", "hwCredentialId" : "9" } '</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

ix). To add a Linux OS:

INPUT	<pre>curl -u admin:admin https://localhost/api /inventory?media=json -X POST -H "content-type: application/json" -d \ ' { "managementLevel" : "FULL", "polledName" : "LINUX OS", "deviceType" : "Server", "osConnMethod" : "oslinuxssh", "osConnPort" : "0", "osCredentialId" : "5" } '</pre>
OUTPUT	<pre>{ "message" : "Queued" }</pre>

GET Method detail - inventory/{deviceId}

List details of an inventory device, in either XML or JSON formats.

Request Parameters

None.

Response data keys

Name	Description
capabilities	device capabilities: routing, switching, switching & routing.
certified	if device has been certified.
cmCredentialId	ID of the config management credential set.
cmCredentialName	name of the config management credential - this name will only be displayed if it is a shared credential (i.e. not a local credential).
configurationManagementLicense	if Configuration Management is enabled ('true') or disabled ('false'), by default false.
context	SNMP v3 context.

credentialId	ID of the shared credential set, used for SNMP and virtualization assets. The credential must be a valid credential for the specified device type. credentialId is based on the credentials you have created and therefore can be any value. E.g. if credentialId = 1 is an SNMP credential, and the device requires an API Key credential, the request will then fail.
deviceType	device type.
dsObjectId	device's stormworks identifier.
id	device's unique identifier.
hwConnMethod	connection method used to the hardware asset, which is the collector name based on the device type being added, from the following: Server: • dellidracssh • hpilossh • idracsnmp • hpinsightsnmp • ciscoucscimcsnmp • openmanagesnmp Storage: • naviseccli • dellisilonssh • nimble • vnxcontrolssh • dellisilonrest • uemcli • netappsnpmp • datadomainsnmp • dellequallogicsnmp • datadomain
hwConnPort	connection port used to the hardware asset.
hwCredentialId	ID of the hardware asset credential set.
managementIP	management IP address.
managementLevel	management level: • FULL • FULL_NO_PORTS • BASIC • PING_ONLY • WEB.
name	device name

nameUsing	device name is determined using either: • CUSTOMNAME • IPADDRESS • POLLEDNAME • RESOLVABLENAME • RESOLVABLENAMEFQ • SYSTEMNAME.
osConnMethod	connection method used to the OS, which is the collector name based on the device type being added, from the following: • osaixssh • ossolarisssh • oslinuxssh • oswinrm
osConnPort	connection port used to the OS.
osCredentialId	ID of the OS credential set.
polledName	DNS name or IP Address.
protocol	Transport protocol: IPv4 or IPv6.
serverId	server identifier.
snmpBulk	whether SNMPBulkGet is used ('true' or 'false', by default false).
snmpPDUSize	maximum size of SNMP PDU, where 0 = system default.
snmpPort	SNMP port number (if omitted, the default value will be used).
snmpRetry	number of SNMP retries, where 0 = system default.
snmpTimeout	SNMP timeout in seconds, where 0 = system default.
snmpType	SNMP version: v1, v2, v3 or v1/2. For a non-SNMP device, this will be 'none'.
sysDescription	SNMP description field.
sysLocation	SNMP retrieved system Location field.
sysOid	SNMP system identifier field.

udConn Method	connection method used to the user defined object.
udConn Port	connection port used to the user defined object.
username	SNMP v3 user name.
webURL	virtual platform URL (non-Amazon virtual platforms).
zoneId	the ID of a zone.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/inventory/2?media=json</code>
--------------	---

OUTPUT	<pre>{ "serverId" : "34564e92-3b4f-43ba-94a8-04309c0e48fe", "id": "2", "name": "10.66.23.1", "dsObjectId": 814, "snmpTimeout": 0, "snmpRetry": 0, "snmpPDUSize": 0, "protocol": "IPv4", "snmpType": "v3", "nameUsing": "CUSTOMNAME", "certified": "Yes", "polledName": "10.66.23.1", "managementIP": "10.66.23.1", "sys0id": ".1.3.6.1.4.1.2.3.50", "sysDescription": "10/100 Mbps Ethernet Switch", "sysLocation": "Simulator", "deviceType": "Ethernet Switch", "readCommunity": "public", "userName": "entuity", "authType": "MD5", "encrType": "DES", "managementLevel": "FULL", "context": "", "zoneId": 0, "cliUsername": "" }</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/inventory/2?media=xml
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <inventoryDevice cliUsername="" zoneId="0" snmpPDUSize="0" snmpRetry="0" snmpTimeout="0" context="" encrType="NONE" authType="NONE" userName="" readCommunity="public" snmpType="v1/v2c" protocol="IPv4" certified="Yes" deviceType="Ethernet Switch" managementLevel="FULL_MGMT_PORT_ONLY" capabilities="Switch" sysLocation="Simulator" sysDescription="10/100 Mbps Ethernet Switch" sysOid=".1.3.6.1.4.1.2.3.50" managementIP="10.66.23.1" polledName="10.66.23.1" nameUsing="CUSTOMNAME" dsObjectId="922" name=" 10.66.23.1" id="2" serverId="34564e92-3b4f-43ba-94a8-04309c0e48fe"/></pre>
---------------	--

PUT Method detail - inventory/{deviceId}

Change the details of an inventory device, in either XML or JSON formats. Please refer to the Entuity Help Center for help and information on [managing assets via the Entuity UI](#) .

Request Parameters

Same as the table for GET above.

Response data

200 OK

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/inventory/2?media=json -X PUT - H "content-type:application/json" -d \ ' { "managementLevel" : "FULL", "protocol" : "IPv4", "snmpType" : "v3", "userName" : "entuity", "authType" : "MD5", "authPass" : "entuity123", "encrType" : "DES", "encrPass" : "entuity123" }</pre>
--------------	--

```
OUTPUT {  
  "serverId": "34564e92-3b4f-43ba-94a8-04309c0e48fe",  
  "id": "2",  
  "name": "10.66.23.1",  
  "dsObjectId": 814,  
  "snmpTimeout": 0,  
  "snmpRetry": 0,  
  "snmpPDUSize": 0,  
  "protocol": "IPv4",  
  "snmpType": "v3",  
  "nameUsing": "CUSTOMNAME",  
  "certified": "Yes",  
  "polledName": "10.66.23.1",  
  "managementIP": "10.66.23.1",  
  "sys0id": ".1.3.6.1.4.1.2.3.50",  
  "sysDescription": "10/100 Mbps Ethernet Switch",  
  "sysLocation": "Simulator",  
  "deviceType": "Ethernet Switch",  
  "readCommunity": "public",  
  "userName": "entuity",  
  "authType": "MD5",  
  "encrType": "DES",  
  "managementLevel": "FULL",  
  "context": "",  
  "zoneId": 0,  
  "cliUsername": ""  
}
```

INPUT	<pre>curl -H -u admin:admin https://localhost/api/inventory/2?media=xml -X PUT -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <inventoryDevice managementLevel="FULL" snmpType="v3" userName="entuity" authType="MD5" authPass="entuity123" encrType="DES" encrPass="entuity123" />'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <inventoryDevice cliUsername="" zonelId="0" snmpPDUSize="0" snmpRetry="0" snmpTimeout="0" context="" encrType="DES" authType="MD5" userName="entuity" readCommunity="public" snmpType="v3" protocol="IPv4" certified="Yes" deviceType="Ethernet Switch" managementLevel=" FULL" capabilities="Switch" sysLocation="Simulator" sysDescription="10/100 Mbps Ethernet Switch" sysOid=".1.3.6.1.4.1.2.3.50" managementIP="10.66.23.1" polledName="10.66.23.1" nameUsing=" CUSTOM" dsObjectId="922" name="10.66.23.1" id="6" serverId="34564e92-3b4f-43ba-94a8- 04309c0e48fe"/></pre>

Changing the credentials of a Cisco UCS server device:

INPUT	<pre>curl -H -u admin:admin https://localhost/api/inventory/14?media=json -X PUT -H "content-type:application/json" -d \ '{ "managementLevel" : "CLI", "polledName" : "CISCO UCS", "deviceType" : "Server", "hwConnMethod" : "ciscoucscimcsnmp", "hwConnPort" : "0", "hwCredentialId" : "6" }'</pre>
--------------	--

```
OUTPUT {  
  "serverId": "{serverId}",  
  "id": "11",  
  "name": "CISCO UCS",  
  "dsObjectId": 1629,  
  "snmpTimeout": 0,  
  "snmpRetry": 0,  
  "snmpPDUSize": 0,  
  "protocol": "IPv4",  
  "snmpType": "None",  
  "nameUsing": "POLLEDNAME",  
  "certified": "",  
  "polledName": "CISCO UCS",  
  "managementIP": "xx.xx.xxx.xxx",  
  "sys0id": "",  
  "sysDescription": "",  
  "sysLocation": "",  
  "deviceType": "Server",  
  "managementLevel": "CLI",  
  "zoneId": 0,  
  "snmpPort": 0,  
  "snmpBulk": false,  
  "hwConnMethod": "ciscoucscimcsnmp",  
  "hwCredentialId": 9,  
  "hwCredentialName": "snmpCreds2",  
  "deviceName": "CISCO UCS",  
  "configManagementLicense": false  
}
```

DELETE Method details - inventory/{deviceId}

Remove a device from the inventory.

Request

The ID of the object that is to be deleted is sent as a part of a URL.

Response

“OK” message if the device was removed successfully, an error message otherwise.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/inventory/2?media=json -X DELETE</code>
OUTPUT	<pre>{ "message": "OK" }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/inventory/2?media=xml -X DELETE</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>

Note, for backwards compatibility, users can still add a device or devices using a local credential (via api /inventory POST or api/inventory/{deviceId} PUT), but the credential details of such devices are not accessible via GET calls. The applicable parameters are as follows:

- authKey, authPass, authType.
- encrKey, encrPass, encrType.
- readCommunity, writeCommunity.
- webAccessKey, webPassword, webPlatformType, webSecretKey, webTenantId, webUser.

Auto Discovery Status

View a summary of the current status of the Auto Discovery, and start a discovery. Applicable to Enuity v19.0 upwards only.

URL: autodiscovery

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
- [POST Method detail](#)
 - [Request Parameters](#)
 - [Response data](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- GET Method - returns a summary of the current status of the Auto Discovery.
- POST Method - given a profile name, will start a discovery with that profile.
- PUT Method - given a profile name, will stop a discovery with that profile.

GET Method detail

Will show a status summary, displaying the current situation of the discovery, such as any running profiles, and any profiles that are queued to run.

Request Parameters

None.

Response data keys

Name	Description
id	ID number for discovery profile
name	user-specified name of discovery profile
description	user-specified description of discovery profile

status	status of discovery profile: • 0 - Never Ran • 1 - In Progress • 2 - Complete • 3 - Stopped (aborted or crashed)
percentage	percentage of completion, 0 - 100

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/autodiscovery?media=json</code>
OUTPUT	<pre>{ "settings" : [{ "id" : 5, "name" : "default", "description" : "Migrated from old discovery: default.cfg", "status" : 3, "percentage" : 0 }, { "id" : 12, "name" : "test", "description" : "", "status" : 2, "percentage" : 100 }] }</pre>

POST Method detail

Will run the given Auto Discovery profile. If this profile is running already, or is in the queue, then this call will do nothing.

Request Parameters

Name	Description
id	ID of the profile to be run.

Response data

Shows the summary if the call was successful.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/autodiscovery?media=json -X POST -H "content-type:application/json" -d \ '{"id" : "1"}'</pre>
OUTPUT	<pre>{ "settings" : [{ "id" : 1, "name" : "Default", "description" : "Migrated from old discovery: Default.cfg", "status" : 1, "percentage" : 0 }] }</pre>

PUT Method details

Will stop the given Auto Discovery profile.

Request

Name	Description
id	ID of the profile to be stopped.

Response

Shows the summary if the call was successful.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/autodiscovery? media=json -X PUT -H "content-type:application/json" -d \ '{"id" : "1"}'</pre>
--------------	--

OUTPUT	<pre>{ "settings" : [{ "id" : 1, "name" : "Default", "description" : "Migrated from old discovery: Default.cfg", "status" : 3, "percentage" : 100 }]</pre>
---------------	--

Auto Discovery Profiles

View and create Auto Discovery profiles. Applicable to Enuity v19.0 upwards only.

URL: autodiscoveryProfiles

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Examples](#)
- [POST Method detail](#)
 - [Request Parameters](#)
 - [Response data](#)
 - [Examples](#)

Method Summary

- GET Method - returns a list of Auto Discovery profiles (XML, JSON).
- POST Method - creates a new Auto Discovery profile.

GET Method detail

Lists all Auto Discovery profiles.

Request Parameters

None.

Response data keys

Name	Description
items	a list of profiles on this server.
count	number of profiles.

Examples

No Auto Discoveries running, no schedules:

INPUT	<code>curl -u admin:admin https://localhost/api/autodiscoveryProfiles?media=json</code>
-------	---

```

OUTPUT {
    "settings" : [ {
        "id" : 1,
        "name" : "Default",
        "description" : "Migrated from old discovery: Default.cfg",
        "status" : 0,
        "percentage" : 0
    }, {
        "id" : 2,
        "name" : "Custom Discovery",
        "description" : "",
        "status" : 2,
        "percentage" : 100
    }, {
        "id" : 3,
        "name" : "checking test",
        "description" : "",
        "status" : 2,
        "percentage" : 100
    } ]
}

```

POST Method detail

Create a new profile with given settings. If some settings are not given, then they will be filled with default values. The "profile" value must be unique.

Request Parameters

Name	Description
id	ID number for the discovery profile.
name	user-specified name of the discovery profile.
description	user-specified description of the discovery profile.
zoneID	zone ID.

includedAddresses	addresses to scan. Comma-separated IP addresses or hostnames.
excludedAddresses	addresses to exclude. Comma-separated IP addresses or hostnames.
flags	a bitflag for flags: • 0 - manage discovered devices after discovery finishes. • 1 - allow duplicate IPs. • 2 - use all credentials. If this is specified, the 'Credentials' field is ignored. • 3 - managed associated devices after discovery. • 5 - use management IP.
credentials	list of credential IDs to try.
scheduleBitSet	a bitflag for the days on which the schedule should be run. If 0, it does not run. 0: Monday to 6: Sunday.
scheduleHours	hour in which the schedule should be run in 24-hour time.
autoManagementLevel	the level of device management, if automanage is on: • 1 - full. • 2 - full, management interface only. • 3 - full, no interfaces. • 4 - basic. • 5 - basic, ping only. • 6 - web. • 7 - none. • 8 - CLI (not used).
displayNameUsing	the name to be used in management: • 0 - system. • 1 - resolved. • 2 - resolved FQ. • 3 - polled name. • 4 - IP address.
pollUsing	the method of polling when managed: • 0 - poll using given address. • 1 - use resolved name. • 2 - use resolved name, fallback to given address.
destinationViewPath	viewPath to which to add, if managed.
viewPathHidden	indicates if the viewPath should be hidden due to permissions. If true, this viewPath will be ignored when editing.
probeBitSet	a bitflag for probes to run.

attributes	custom attributes: • tcp_ports • udp_ports • snmp.ports • snmp.excluded_sysoids • winrm.ports • ssh.ports
lastFinished	the time a discovery finished.
nextRun	the time a discovery is next scheduled to run.
lastRunStatus	the status of the last run: • 0 or -1 - never ran. • 1 - in progress. • 2 - complete. • 3 - stopped.
credentialStatus	not used.
accessMethods	not used.
lastRan	the time a discovery last ran.
status	current status of this discovery profile: • 0 - never ran. • 1 - in progress. • 2 - complete. • 3 - stopped.
percentage	current progress percentage of the discovery, if running.

Response data

Displays the list of available profiles, updated with the new profile.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/autodiscoveryProfiles -X POST -H 'content-type:application/json' -d \</code>
OUTPUT	<pre>{ "id" : 2, "name" : "Checking poseidon", "description" : "", "zoneId" : 0, "includedAddresses" : "poseidon",</pre>

```

"excludedAddresses" : "",
"flags" : 0,
"publicServicesBitSet" : 0,
"credentials" : [ 7 ],
"scheduleBitSet" : 0,
"scheduleHours" : 0,
"autoManagementLevel" : 1,
"displayNameUsing" : 5,
"pollUsing" : 2,
"destinationViewPath" : null,
"viewPathHidden" : null,
"probeBitSet" : 30,
"attributes" : {
"collectors" : "oswinrm:oslinuxssh",
"http.ports" : "80",
"https.ports" : "443",
"snmp.excluded_sysoids" : "1.3.6.1.4.1.311.1.1.3.1.2,\n1.
3.6.1.4.1.2.3.1.2.1.1.3,\n1.3.6.1.4.1.8072.3.2.10,\n1.
3.6.1.4.1.8072.3.2.3,\n1.3.6.1.4.1.311.1.1.3.1.1,\n1.
3.6.1.4.1.311.1.1.3.1.3,\n1.3.6.1.4.1.42.2.1.1",
"snmp.ports" : "161",
"ssh.ports" : "22",
"tcp_ports" : "21, 23, 25",
"wirnm.ports" : "5985, 5986"
},
"lastFinished" : null,
"nextRun" : null,
"lastRunStatus" : null,
"credentialStatus" : null,
"accessMethods" : [ "Host Detection", "SNMP", "WinRM", "SSH" ],
"lastRan" : null,
"status" : null,
"percentage" : null
testuser@testlaptop:/mnt/c/Users/testuser curl -sku admin:
admin'https://localhost/api/autodiscoveryProfiles'
{
"settings" : [ {
"id" : 1,

```

```
"name" : "Default",  
"description" : "Migrated from old discovery: Default.cfg",  
"status" : 2,  
"percentage" : 100  
}, {  
"id" : 2,  
"name" : "Checking poseidon",  
"description" : "",  
"status" : 0,  
"percentage" : 0  
} ]  
}
```

Auto Discovery Profile

View, edit and delete Auto Discovery Profiles. Applicable to Enuity v19.0 upwards only.

URL: autodiscoveryProfiles/{id}

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Examples](#)
- [PUT Method detail](#)
 - [Request Parameters](#)
 - [Response data](#)
 - [Examples](#)
- [DELETE Method detail](#)
 - [Request Parameters](#)
 - [Response data](#)
 - [Examples](#)

Method Summary

- GET Method - returns the settings of this profile (XML, JSON).
- PUT Method - modifies this profile's settings.
- DELETE Method - removes this profile.

GET Method detail

Shows the settings of this profile.

Request Parameters

None.

Response data keys

Shows the settings of the given profile.

Name	Description
id	ID number for the discovery profile.
name	user-specified name of the discovery profile.
description	user-specified description of the discovery profile.
zoneID	zone ID.

includedAddresses	addresses to scan. Comma-separated IP addresses or hostnames.
excludedAddresses	addresses to exclude. Comma-separated IP addresses or hostnames.
flags	a bitflag for flags: • 0 - manage discovered devices after discovery finishes. • 1 - allow duplicate IPs. • 2 - use all credentials. If this is specified, the 'Credentials' field is ignored. • 3 - managed associated devices after discovery. • 5 - use management IP.
credentials	list of credential IDs to try.
scheduleBitSet	a bitflag for the days on which the schedule should be run. If 0, it does not run. 0: Monday to 6: Sunday.
scheduleHours	hour in which the schedule should be run in 24-hour time.
autoManagementLevel	the level of device management, if automanage is on: • 1 - full. • 2 - full, management interface only. • 3 - full, no interfaces. • 4 - basic. • 5 - basic, ping only. • 6 - web. • 7 - none. • 8 - CLI (not used).
displayNameUsing	the name to be used in management: • 0 - system. • 1 - resolved. • 2 - resolved FQ. • 3 - polled name. • 4 - IP address.
pollUsing	the method of polling when managed: • 0 - poll using given address. • 1 - use resolved name. • 2 - use resolved name, fallback to given address.
destinationViewPath	viewPath to which to add, if managed.
viewPathHidden	indicates if the viewPath should be hidden due to permissions. If true, this viewPath will be ignored when editing.

probeBitSet	a bitflag for probes to run: • 0 - All. Will override the below flags. • 1 - Host detection (ping) • 2 - SNMP • 3 - WinRM • 4 - SSH • 5 - HTTP (not in use) • 6 - HTTPS (not in use) • 7 - DataDomain • 8 - iDrac SNMP • 9 - HP iLO SSH • 10 - iDrac SNMP • 11 - navisecli • 12 - navisecli SSH • 13 - NetApp SSH • 14 - Nimble • 15 - uemcli • 16 - VNX Control SSH • 17 - Linux (Collection Engine) • 18 - WinRM (Collection Engine)
attributes	custom attributes: • tcp_ports • udp_ports • snmp.ports • snmp.excluded_sysoids • winrm.ports • ssh.ports
lastFinished	the time a discovery finished.
nextRun	the time a discovery is next scheduled to run.
lastRunStatus	the status of the last run: • 0 or -1 - never ran. • 1 - in progress. • 2 - complete. • 3 - stopped.
credentialStatus	not used.
accessMethods	access method used, choosing from the following: • Host Detection • WinRM • SSH • SNMP
lastRan	the time a discovery last ran.

status	current status of this discovery profile: • 0 - never ran. • 1 - in progress. • 2 - complete. • 3 - stopped.
percentage	current progress percentage of the discovery, if running.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/autodiscoveryProfiles/5? media=json</code>
--------------	--

```

OUTPUT {
  "id" : 5,
  "name" : "default",
  "description" : "migrated from old discovery: default.cfg",
  "zoneId" : 0,
  "includedAddresses" : "10.55.0-255.0-255",
  "excludedAddresses" : "",
  "flags" : 6,
  "publicServicesBitSet" : 0,
  "credentials" : null,
  "scheduleBitSet" : 1
  "scheduleHours" : 7,
  "autoManagementLevel" : 1,
  "displayNameUsing" : 3,
  "pollUsing" : 1,
  "destinationViewPath" : null,
  "viewPathHidden" : false,
  "probeBitSet" : 6,
  "attributes" : {
    "snmp.excluded_sysoids" :
    "1.3.6.1.4.1.311.1.1.3.1.2,1.3.6.1.4.1.2.3.1.2.1.1.3,1.3.6.1.4.1.8072.3.2.
    10,1.3.6.1.4.1.8072.3.2.3,1.3.6.1.4.1.311.1.1.3.1.1,1.3.6.1.4.1.311.1.1.3.
    1.3,1.3.6.1.4.1.42.2.1.1",
    "snmp.ports" : "161",
    "ssh.ports" : "22",
    "tcp_ports" : "21, 23, 25",
    "winrm.ports" : "5985, 5986"
  },
  "lastFinished" : 1617295096000,
  "nextRun" : 1618207200000,
  "lastRunStatus" : -1,
  "credentalStatus" : null,
  "accessMethods" : [ "Host Detection", "SNMP" ],
  "lastRan" : 1617295095000,
  "status" : 3,
  "percentage" : 0
}

```

INPUT	curl -u admin:admin https://localhost/api/autodiscoveryProfiles/2?media=xml
OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:autoDiscProfile autoManagementLevel="1" description="10.44.XX.XX" destinationViewPath="My Network/local/10.44.XX.XX" displayNameUsing="5" excludedAddresses="" flags="41" id="2" includedAddresses="10.44.0-255.0- 255" lastFinished="1662565889000" lastRan="1662565869000" lastRunStatus=" -1" name="10.44.XX.XX" percentage="0" pollUsing="2" probeBitSet="30" publicServicesBitSet="0" scheduleBitSet="0" scheduleHours="0" status="3" viewPathHidden="false" zoneId="0" xmlns:ns5=" http://www.entuity.com/webrpc " xmlns:ns2=" http://www.entuity.com/schemas/webUI" xmlns:ns4=" http://www. entuity.com/schemas/eventengine" xmlns:ns3=" http://www.entuity.com/schemas /flow"> <accessMethods> <accessMethod>Host Detection</accessMethod> <accessMethod>SNMP</accessMethod> <accessMethod>WinRM</accessMethod> <accessMethod>SSH</accessMethod> </accessMethods> <attributes> <entries> <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">https.ports> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">443> </entries> <entries> <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">snmp. excluded_sysoids> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1. 3.6.1.4.1.311.1.1.3.1.2, 1.3.6.1.4.1.2.3.1.2.1.1.3, 1.3.6.1.4.1.8072.3.2.10, 1.3.6.1.4.1.8072.3.2.3, 1.3.6.1.4.1.311.1.1.3.1.1, 1.3.6.1.4.1.311.1.1.3.1.3, 1.3.6.1.4.1.42.2.1.1 </value> </entries> <entries> <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">collectors> </pre>

```

    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">oswinrm:
oslinuxssh:ossolarisssh:osaixssh:ciscoucscimcsnmp:idracsntp:dellidracssh:
hpinsightsntp:hpiiosssh:openmanagesntp:supermicrorest:datadomainsntp:
datadomain:naviseccli:vnxcntrlssh:uemcli:dellequallogicsntp:
dellisilonrest:dellisilonssh:netappsnmp:netappssh:nimble:purestoragerest>
  </entries>
<entries>
  <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">ssh.ports>
    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">22>
  </entries>
<entries>
  <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">restapi.ports>
    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">80>
  </entries>
<entries>
  <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">http.ports>
    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">80>
  </entries>
<entries>
  <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">snmp.ports>
    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">161>
  </entries>
<entries>
  <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">winnm.ports>
    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">5985,
5986</value>
  </entries>
<entries>
  <key xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">tcp_ports>
    <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">21, 23,
25</value>

```

```

    </entries>
  </attributes>
  <credentials>
    <credential>1</credential>
    <credential>2</credential>
    <credential>3</credential>
    <credential>4</credential>
    <credential>5</credential>
    <credential>6</credential>
    <credential>8</credential>
    <credential>9</credential>
    <credential>10</credential>
    <credential>11</credential>
    <credential>12</credential>
  </credentials>
</ns2:autoDiscProfile>

```

PUT Method detail

Modifies the chosen profile with given settings. The name cannot be modified.

Request Parameters

Name	Description
id	ID number for the discovery profile.
name	user-specified name of the discovery profile.
description	user-specified description of the discovery profile.
zoneID	zone ID.
includedAddresses	addresses to scan. Comma-separated IP addresses or hostnames.
excludedAddresses	addresses to exclude. Comma-separated IP addresses or hostnames.

flags	a bitflag for flags: • 0 - manage discovered devices after discovery finishes. • 1 - allow duplicate IPs. • 2 - use all credentials. If this is specified, the 'Credentials' field is ignored. • 3 - managed associated devices after discovery. • 5 - use management IP.
credentials	list of credential IDs to try.
scheduleBitSet	a bitflag for the days on which the schedule should be run. If 0, it does not run. 0: Monday to 6: Sunday.
scheduleHours	hour in which the schedule should be run in 24-hour time.
autoManagementLevel	the level of device management, if automanage is on: • 1 - full. • 2 - full, management interface only. • 3 - full, no interfaces. • 4 - basic. • 5 - basic, ping only. • 6 - web. • 7 - none. • 8 - CLI (not used).
displayNameUsing	the name to be used in management: • 0 - system. • 1 - resolved. • 2 - resolved FQ. • 3 - polled name. • 4 - IP address.
pollUsing	the method of polling when managed: • 0 - poll using given address. • 1 - use resolved name. • 2 - use resolved name, fallback to given address.
destinationViewPath	viewPath to which to add, if managed.
viewPathHidden	indicates if the viewPath should be hidden due to permissions. If true, this viewPath will be ignored when editing.
probeBitSet	a bitflag for probes to run.

attributes	custom attributes: • tcp_ports • udp_ports • snmp.ports • snmp.excluded_sysoids • winrm.ports • ssh.ports
lastFinished	the time a discovery finished.
nextRun	the time a discovery is next scheduled to run.
lastRunStatus	the status of the last run: • 0 or -1 - never ran. • 1 - in progress. • 2 - complete. • 3 - stopped.
credentialStatus	not used.
accessMethods	not used.
lastRan	the time a discovery last ran.
status	current status of this discovery profile: • 0 - never ran. • 1 - in progress. • 2 - complete. • 3 - stopped.
percentage	current progress percentage of the discovery, if running.

Response data

Shows the settings of the added profile. Uses the parameters shown above.

Examples

Modifying managementLevel:

INPUT	<pre>curl -u admin:admin https://localhost/api/autodiscoveryProfiles/5? media=json -X PUT -H "content-type:application/json" -d \ '{"autoManagementLevel" : "5"}'</pre>
OUTPUT	<pre>{ "id" : "5", "name" : "default", "description" : "migrated from old discovery: default.cfg",</pre>

```

"zoneId" : 0
"includedAddresses" : "10.55.0-255.0-255",
"excludedAddresses" : "",
"flags" : 6,
"publicServicesBitSet" : 0,
"credentials" : null,
"scheduleBitSet" : 1
"scheduleHours" : 7,
"autoManagementLevel" : 5,
"displayNameUsing" : 3,
"pollUsing" : 1,
"destinationViewPath" : null,
"viewPathHidden" : false,
"probeBitSet" : 6,
"attributes" : {
  "snmp.excluded_sysoids" :
"1.3.6.1.4.1.311.1.1.3.1.2,1.3.6.1.4.1.2.3.1.2.1.1.3,1.3.6.1.4.1.8072.3.2.
10,1.3.6.1.4.1.8072.3.2.3,1.3.6.1.4.1.311.1.1.3.1.1,1.3.6.1.4.1.311.1.1.3.
1.3,1.3.6.1.4.1.42.2.1.1",
  "snmp.ports" : "161",
  "ssh.ports" : "22",
  "tcp_ports" : "21, 23, 25",
  "winrm.ports" : "5985, 5986"
},
"lastFinished" : 1617295096000,
"nextRun" : 1618207200000,
"lastRunStatus" : -1,
"credentalStatus" : null,
"accessMethods" : [ "Host Detection", "SNMP" ],
"lastRan" : 1617295095000,
"status" : 3,
"percentage" : 0
}

```

DELETE Method detail

Removes the given profile.

Request Parameters

None.

Response data

None.

Examples

Returns an OK code if successful.

Auto Discovery Results

View the results of all Auto Discovery profiles. Applicable to Enuity v19.0 upwards only.

URL: autodiscoveryResults

- [Method Summary](#)
- [GET Method detail](#)
 - [Examples](#)

Method Summary

- GET Method - lists all discovery results.

GET Method detail

Lists all discovery results.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/autodiscoveryResults?media=json</code>
-------	--

```

OUTPUT {
  "assets" : [ {
    "id" : 1,
    "parentId" : 0,
    "profileId" : 2,
    "flags" : 0,
    "deviceName" : "madrid-routerb",
    "ipAddr" : "10.200.44.1",
    "deviceType" : null,
    "description" : "Cisco IOS Software, 3600 Software (C3640-A3JS-M),
Version 12.4(25b), RELEASE SOFTWARE (fc1)\r\nTechnical Support: https://www
.cisco.com/techsupport\r\nCopyright (c) 1986-2009 by Cisco Systems, Inc.
\r\nCompiled Wed 12-Aug-09 12:52 by prod_rel_team",
    "firstSeenSec" : 1617096367,
    "lastSeenSec" : 1617109294,
    "credentialIds" : {
      "0" : 1,
      "2" : 1
    },
    "credentialStatus" : 2,
    "vendor" : null,
    "model" : null,
    "serialNo" : null,
    "location" : "Server Room - Cabinet B",
    "family" : null,
    "managementProtocol" : 2,
    "managementOptions" : "p=2:c=1:r=161",
    "assetCategory" : 1,
    "hypervisor" : null,
    "zoneId" : 0,
    "newAsset" : false,
    "managed" : true,
    "configManagement" : null,
    "managedSinceSec" : 1617097338,
    "managementLevel" : 1
  }, {

```

INPUT	curl -u admin:admin https://localhost/api/autodiscoveryResults?media=xml
OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <autoDiscoveryResults xmlns:ns5="http://www.entuity.com/webrpc" xmlns:ns2=" http://www.entuity.com/schemas/webUI" xmlns:ns4="http://www.entuity.com /schemas/eventengine" xmlns:ns3="http://www.entuity.com/schemas/flow"> <assets> <assets assetCategory="1" credentialStatus="0" deviceName="mxg.entuity. local" firstSeenSec="1662542213" flags="0" id="1" ipAddr="10.44.1.110" lastSeenSec="1662542213" managed="false" managementProtocol="0" newAsset=" true" parentId="0" profileId="2" zoneId="0"/> <assets assetCategory="1" credentialStatus="0" deviceName="10.44.1.93" firstSeenSec="1662542213" flags="0" id="2" ipAddr="10.44.1.93" lastSeenSec="1662542213" managed="false" managementProtocol="0" newAsset=" true" parentId="0" profileId="2" zoneId="0"/> <assets assetCategory="1" credentialStatus="1" deviceName="sky.entuity. local" firstSeenSec="1662542213" flags="0" id="3" ipAddr="10.44.1.23" lastSeenSec="1662542213" managed="false" managementProtocol="0" newAsset=" true" parentId="0" profileId="2" zoneId="0"/> <assets assetCategory="1" credentialStatus="1" deviceName="dustdevil. entuity.local" firstSeenSec="1662542213" flags="0" id="4" ipAddr=" 10.44.1.95" lastSeenSec="1662542213" managed="false" managementProtocol=" 0" newAsset="true" parentId="0" profileId="2" zoneId="0"/> <assets assetCategory="1" credentialStatus="1" deviceName="lin64build. entuity.local" firstSeenSec="1662542213" flags="0" id="5" ipAddr=" 10.44.1.11" lastSeenSec="1662542213" managed="false" managementProtocol=" 0" newAsset="true" parentId="0" profileId="2" zoneId="0"/> <assets assetCategory="1" credentialStatus="2" description="Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 12.2(58)SE2, RELEASE SOFTWARE (fc1)&#13;&#10;Technical Support: http://www.cisco.com /techsupport&#13;&#10;Copyright (c) 1986-2011 by Cisco Systems, Inc.&#13; &#10;Compiled Thu 21-Jul-11 02:13 by prod_rel_team" deviceName="bottom2960. entuity.local" firstSeenSec="1662542213" flags="0" id="6" ipAddr=" 10.44.1.41" lastSeenSec="1662542213" location="Server Room - Cabinet A" managed="true" managedSinceSec="1662543949" managementLevel="1" managementOptions="p=2:c=1:r=161" managementProtocol="2" newAsset="false" parentId="0" profileId="2" zoneId="0"> <credentialIds> <entries> <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0> <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1> </entries> <entries> <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2> <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1> </pre>

```

    </entries>
  </credentialIds>
</assets>

  <assets assetCategory="1" credentialStatus="1" deviceName="kb.entuity.
com" firstSeenSec="1662542213" flags="0" id="7" ipAddr="10.44.1.16"
lastSeenSec="1662542213" managed="false" managementProtocol="0" newAsset="
true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="1" deviceName="entlonsw02.
entuity.local" firstSeenSec="1662542213" flags="0" id="8" ipAddr="
10.44.1.25" lastSeenSec="1662542213" managed="false" managementProtocol="
0" newAsset="true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="1" deviceName="entlonsw01.
entuity.local" firstSeenSec="1662542213" flags="0" id="9" ipAddr="
10.44.1.27" lastSeenSec="1662542213" managed="false" managementProtocol="
0" newAsset="true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="2" description="Cisco IOS
Software, C3550 Software (C3550-IPSERVICESK9-M), Version 12.2(44)SE6,
RELEASE SOFTWARE (fc1)&#13;&#10;Copyright (c) 1986-2009 by Cisco Systems,
Inc.&#13;&#10;Compiled Mon 09-Mar-09 20:28 by gereddy" deviceName="top3550.
entuity.local" firstSeenSec="1662542213" flags="0" id="10" ipAddr="
10.44.1.42" lastSeenSec="1662542213" location="Server Room - Cabinet A"
managed="true" managedSinceSec="1662543964" managementLevel="1"
managementOptions="p=2:c=2:r=161" managementProtocol="2" newAsset="false"
parentId="0" profileId="2" zoneId="0">

    <credentialIds>
      <entries>
        <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0>
          <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2>
        </entries>
      <entries>
        <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2>
          <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2>
        </entries>
      </credentialIds>
    </assets>

  <assets assetCategory="1" credentialStatus="2" description="Cisco IOS
Software, C3550 Software (C3550-IPSERVICESK9-M), Version 12.2(44)SE6,
RELEASE SOFTWARE (fc1)&#13;&#10;Copyright (c) 1986-2009 by Cisco Systems,
Inc.&#13;&#10;Compiled Mon 09-Mar-09 20:28 by gereddy" deviceName="
bottom3550.entuity.local" firstSeenSec="1662542213" flags="0" id="11"
ipAddr="10.44.1.12" lastSeenSec="1662542213" location="Server Room -
Cabinet A" managed="true" managedSinceSec="1662543978" managementLevel="1"
managementOptions="p=2:c=1:r=161" managementProtocol="2" newAsset="false"
parentId="0" profileId="2" zoneId="0">

```

```

    <credentialIds>
      <entries>
        <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0>
          <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>
        </entries>
      </entries>
    </credentialIds>
  </assets>

  <assets assetCategory="1" credentialStatus="2" description="Cisco IOS
Software, C3750 Software (C3750-IPSERVICESK9-M), Version 12.2(55)SE9,
RELEASE SOFTWARE (fc1)&#13;&#10;Technical Support: http://www.cisco.com
/techsupport&#13;&#10;Copyright (c) 1986-2014 by Cisco Systems, Inc.&#13;
&#10;Compiled Mon 03-Mar-14 22:45 by prod_rel_team" deviceName="stack3750.
entuity.local" firstSeenSec="1662542213" flags="0" id="12" ipAddr="
10.44.1.9" lastSeenSec="1662542213" location="Server Room - Server Rack"
managed="true" managedSinceSec="1662543993" managementLevel="1"
managementOptions="p=2:c=1:r=161" managementProtocol="2" newAsset="false"
parentId="0" profileId="2" zoneId="0">
    <credentialIds>
      <entries>
        <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0>
          <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>
        </entries>
      </entries>
    </credentialIds>
  </assets>
  ...
[abbreviated]
  ...

```

```

    <assets assetCategory="1" credentialStatus="1" deviceName="tornado.
    entuity.local" firstSeenSec="1662542227" flags="0" id="18" ipAddr="
    10.44.1.136" lastSeenSec="1662542227" managed="false" managementProtocol="
    0" newAsset="true" parentId="0" profileId="2" zoneId="0"/>

    <assets assetCategory="1" credentialStatus="2" description="Cisco IOS
    Software, C2600 Software (C2600-ADVIPSERVICESK9-M), Version 12.4(19),
    RELEASE SOFTWARE (fc1)&#13;&#10;Technical Support: http://www.cisco.com
    /techsupport&#13;&#10;Copyright (c) 1986-2008 by Cisco Systems, Inc.&#13;
    &#10;Compiled Fri 29-Feb-08 19:23 by prod_rel_team" deviceName="r2610.
    entuity.local" firstSeenSec="1662542213" flags="0" id="19" ipAddr="
    10.44.1.35" lastSeenSec="1662542213" location="Server-Room Cabinet A"
    managed="true" managedSinceSec="1662544036" managementLevel="1"
    managementOptions="p=2:c=1:r=161" managementProtocol="2" newAsset="false"
    parentId="0" profileId="2" zoneId="0">

      <credentialIds>

        <entries>

          <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
          " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0>

            <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
            /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>

          </entries>

        <entries>

          <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
          " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2>

            <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
            /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>

          </entries>

        </credentialIds>

      </assets>

      <assets assetCategory="1" credentialStatus="2" description="Cisco IOS
      Software, 2800 Software (C2800NM-ADVENTERPRISEK9-M), Version 12.4(6)XT,
      RELEASE SOFTWARE (fc4)&#13;&#10;Technical Support: http://www.cisco.com
      /techsupport&#13;&#10;Copyright (c) 1986-2007 by Cisco Systems, Inc.&#13;
      &#10;Compiled Thu 18-Jan-07 22:42 by hqluong" deviceName="10.2.11.11"
      firstSeenSec="1662542213" flags="0" id="20" ipAddr="10.2.11.11"
      lastSeenSec="1662542213" location="wonderland" managed="true"
      managedSinceSec="1662544038" managementLevel="1" managementOptions="p=2:
      c=1:r=161" managementProtocol="2" newAsset="false" parentId="0" profileId="
      2" zoneId="0">

        <credentialIds>

          <entries>

            <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
            " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0>

              <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
              /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>

            </entries>

          <entries>

```

```

    <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2>

    <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>

  </entries>

</credentialIds>

</assets>

  <assets assetCategory="1" credentialStatus="1" deviceName="pete-demo.
entuity.local" firstSeenSec="1662542213" flags="0" id="21" ipAddr="
10.44.1.104" lastSeenSec="1662542213" managed="false" managementProtocol="
0" newAsset="true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="1" deviceName="10.44.2.33"
firstSeenSec="1662542242" flags="0" id="22" ipAddr="10.44.2.33"
lastSeenSec="1662542242" managed="false" managementProtocol="0" newAsset="
true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="0" deviceName="10.44.2.34"
firstSeenSec="1662542242" flags="0" id="23" ipAddr="10.44.2.34"
lastSeenSec="1662542242" managed="false" managementProtocol="0" newAsset="
true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="0" deviceName="SYN-ENT-01"
firstSeenSec="1662542242" flags="0" id="24" ipAddr="10.44.2.35"
lastSeenSec="1662542242" managed="false" managementProtocol="0" newAsset="
true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="0" deviceName="sulphur.
entuity.local" firstSeenSec="1662542242" flags="0" id="25" ipAddr="
10.44.2.122" lastSeenSec="1662542242" managed="false" managementProtocol="
0" newAsset="true" parentId="0" profileId="2" zoneId="0"/>

  <assets assetCategory="1" credentialStatus="2" description="APC Web
/SNMP Management Card (MB:v3.8.6 PF:v5.1.3 PN:apc_hw05_aos_513.bin AF1:v5.
1.3 AN1:apc_hw05_sumx_513.bin MN:AP9630 HR:05 SN: ZA1109016383 MD:03/01
/2011) (Embedded PowerNet SNMP Agent SW v2.2 compatible)" deviceName="
10.44.1.65" firstSeenSec="1662542213" flags="0" id="26" ipAddr="
10.44.1.65" lastSeenSec="1662542213" location="London" managed="true"
managedSinceSec="1662544040" managementLevel="1" managementOptions="p=2:
c=1:r=161" managementProtocol="2" newAsset="false" parentId="0" profileId="
2" zoneId="0">

  ...

[abbreviated]

  ...

  <credentialIds>

    <entries>

      <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">0>

      <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>

    </entries>

    <entries>

```

```
        <key xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">2>
        <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001
/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">1>
    </entries>
</credentialIds>
</assets>
</assets>
</autoDiscoveryResults>
```

Auto Discovery Result

View the results of a specific Auto Discovery profile. Applicable to Enuity v19.0 upwards only.

URL: autodiscoveryResults/{id}

- [Method Summary](#)
- [GET Method detail](#)
 - [Examples](#)

Method Summary

- GET Method - lists the results of a specific Auto Discovery profile.

GET Method detail

Lists the results of a specific Auto Discovery profile.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/autodiscoveryResults/5? media=json</code>
-------	---

```
OUTPUT {
  "assets" : [ {
    "id" : 5,
    "parentId" : 0,
    "profileId" : 12,
    "flags" : 0,
    "devicename" : "one.one.one.one",
    "ipAddr" : "1.1.1.1",
    "deviceType" : null,
    "description" : null,
    "firstSeenSec" : 1617720151,
    "lastSeenSec" : 1617720151,
    "credentialIds" : null,
    "credentialStatus" : 0,
    "vendor" : null,
    "model" : null,
    "serialNo" : null,
    "location" : null,
    "family" : null,
    "managementProtocol" : 0,
    "managementOptions" : null,
    "assetCategory" : 1,
    "hypervisor" : null,
    "zoneId" : 0,
    "newAsset" : true,
    "managed" : false,
    "configManagement" : null,
    "managedSinceSec" : null,
    "managementLevel" : null
  } ]
}
```

Product Info

Details information about the installed version of Entuity/ENA.

URL: info

- [Method Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- GET Method - returns information about the installed version of Entuity/ENA.

GET Method detail

Returns information about the installed version of Entuity/ENA.

Response

Response is an object with the following attributes:

Name	Description
hostAddress	host name for accessing the product.
id	server ID.
product	product edition.
sslAccess	specifies to use HTTP (false) or HTTPS (true)
version	product version.
versionDisplay	user-friendly version string of the product.
webPort	port number for accessing the product over HTTP(S).

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/info?media=json</code>
-------	--

OUTPUT	<pre>{ "id": "34564e92-3b4f-43ba-94a8-04309c0e48fe", "version": "17.0.0.p0", "versionDisplay": "Entuity 17.0", "product": "EYE", "hostAddress": "ENTLONDEV08", "webPort": 80, "sslAccess": false }</pre>
---------------	--

INPUT	<code>curl -u admin:admin https://localhost/api/info?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <serverInfo sslAccess="false" webPort="80" hostAddress="ENTLONDEV08" product="EYE" versionDisplay="Entuity 17.0" version="17.0.0.p0" id=" 34564e92-3b4f-43ba-94a8-04309c0e48fe"/></pre>

Servers

List servers.

URL: servers

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Examples](#)

Method Summary

- GET Method - lists Entuity servers (XML, JSON).

GET Method detail

Lists Entuity servers, in either XML or JSON format.

Request Parameters

None

Response data keys

Name	Description
count	number of servers.
servers	list of servers.
server	Server summary details, with the following attributes: • hostname: DNS name of the host • Id: Unique server Identifier.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/servers?media=json</code>
-------	---

OUTPUT	<pre>{ "items": [{ "serverId": "34564e92-3b4f-43ba-94a8-04309c0e48fe", "id": "34564e92-3b4f-43ba-94a8-04309c0e48fe", "name": "ENTLONDEV08" }], "count": 1 }</pre>
---------------	---

INPUT	<code>curl -u admin:admin https://localhost/api/servers?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="namedItem" name="ENTLONDEV08" id="34564e92-3b4f-43ba-94a8-04309c0e48fe" serverId="34564e92-3b4f-43ba-94a8-04309c0e48fe" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>

Servers Details

List details of a server.

URL: servers/{serverId}

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Examples](#)

Method Summary

- GET Method - list details of a server (XML, JSON).

GET Method detail

List details of a server, in either XML or JSON format.

Request Parameters

None

Response data keys

Name	Description
centralServer	if the server is a Central Server: True or False.
hostname	name of host running the Entuity server.
included	if the server provides results in a multi server system: True or False.
licensed	if the server is licensed: True or False.
local	if this server is the one servicing the request: True or False
role	role of the server: Polling, FlowCollector, ESPServer.
serverId	unique server identifier.
ssl	if the server is configured to use Secure Socket Layer: True or False.
webPort	port number of the web server.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/server/34564e92-3b4f-43ba-94a8-04309c0e48fe?media=json</code>
OUTPUT	<pre>{ "name": "ENTLONDEV08", "included": true, "role": "Polling", "webPort": 80, "ssl": false, "local": true, "serverId": "34564e92-3b4f-43ba-94a8-04309c0e48fe", "licensed": true, "centralServer": false }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/servers/34564e92-3b4f-43ba-94a8-04309c0e48fe?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <entuityServer licensed="true" role="Polling" local="true" included=" true" centralServer="false" ssl="false" webPort="80" name="ENTLONDEV08" serverId="34564e92-3b4f-43ba-94a8-04309c0e48fe"/></pre>

User Groups

This resource lists general information about user groups.

URL: userGroups

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists available user groups.
- POST Method - creates a new user group.

GET Method detail

Lists available user groups. For non-administrators, the list is restricted to the groups to which the user currently belongs.

Response

Response includes a list of user groups. Each user group has the following attributes:

Name	Description
id	user group id unique to the server.
name	user group name.
serverId	Entuity Server Id on which resource resides.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/userGroups?media=json</code>
-------	--

OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "1", "name": "Administrators" }, { "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "2", "name": "All Users" }], "count": 2 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/userGroups?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="Administrators" id="1" serverId=" ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"/> <item xsi:type="namedItem" name="All Users" id="2" serverId="ee934fe2- 1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>

POST Method details

Creates a new group of users.

Request

Name	Description
Name	name of the group.

Response

The updated list of user groups, as the GET method would return them.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/userGroups?media=json -X POST -H "content-type:application/xml" -d \ '{' "name" : "Support" }'</pre>
OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "1", "name": "Administrators" }, { "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "2", "name": "All Users" }, { "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "6", "name": "Support" }], "count": 3 }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/userGroups?media=xml -X POST -H "content-type:application/xml" -d \ '<userGroupInfo audit="false"> <name>Support</name> </userGroupInfo>'</pre>
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="3"> <item xsi:type="namedItem" name="Administrators" id="1" serverId=" ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"/> <item xsi:type="namedItem" name="All Users" id="2" serverId="ee934fe2- 1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> <item xsi:type="namedItem" name="Support" id="6" serverId="ee934fe2- 1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>
---------------	--

User Group Details

Lists a set of operations on a particular group.

URL: `userGroups/{groupId}`

- [Methods Summary](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- DELETE Method - deletes the user group.

DELETE Method details

Deletes the user group.

Response

The updated list of user groups, as GET method would return them.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/userGroups/3?media=json -X DELETE</code>
-------	--

OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "1", "name": "Administrators" }, { "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "2", "name": "All Users" }], "count": 2 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/userGroups/3?media=xml -X DELETE
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="Administrators" id="1" serverId=" ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"/> <item xsi:type="namedItem" name="All Users" id="2" serverId="ee934fe2- 1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>

User Group Permissions

Lists and modifies the set of tools accessible to a user group.

URL: `userGroups/{groupID}/tools`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists tools that this user group has permission to access.
- PUT Method - modifies the list of tools that this user group has permission to access.

GET Method details

The list of tools that this group has a permission to access.

Response

List of items, each one according to the following format:

Name	Description
id	tool ID.
name	tool name.
subgroup	the name of subgroup that given tool is a part of.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/userGroups/6/tools?media=json</code>
-------	--

OUTPUT	<pre>{ "items": [], "count": 0 }</pre>
---------------	--

INPUT	<code>curl -u admin:admin https://localhost/api/userGroups/6/tools?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="0"/></pre>

PUT Method details

Modifies the list of tools to which the user group has permission to access.

Request

The list of tool IDs.

Response

The updated list of group permissions, as the GET method would return them.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/userGroups/6/tools?media=json X PUT -H "content-type:application/json" -d \ { "tools" : [{ "id" : 15 }, { "id" : 89 }] }</pre>
--------------	---

OUTPUT	<pre>{ "items" : [{ "id" : 15, "toolName" : "Ticker", "groupName" : "Tools" }, { "id" : 89, "toolName" : "Data Export", "groupName" : "Administrator Tools" }], "count" : 2 }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin https://localhost/api/userGroups/6?media=xml -X PUT - H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <groupToolsWrapper> <tool id="15" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> <tool id="89" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </groupToolsWrapper>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="toolId" groupName="Tools" id="15" toolName="Ticker" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="toolId" groupName="Administrator Tools" id="89" toolName="Data Export" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>

Tools

Lists tools available on the server.

URL: tools

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Methods - lists all tools available on the server.

GET Method details

Lists all tools available on the server.

Response

The list of tools are grouped into subgroups, where every entry has the following format:

Method	Description
id	tool ID.
name	the name of the tool.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/tools?media=json</code>
-------	---

OUTPUT	<pre>{ "items" : [{ "subgroup" : "Administrator Tools", "tools" : [{ "id" : 79, "name" : "Event Administration" }, { ...removed for brevity... }, { "id" : 111, "name" : "UD Polling" }] }, { ...removed for brevity... }, { "subgroup" : "Tools", "tools" : [{ "id" : 15, "name" : "Ticker" }, { ...removed for brevity... }, { "id" : 112, "name" : "Configuration Management" }] ...removed for brevity... }], "count" : 6 }</pre>
---------------	---

INPUT	<code>curl -u admin:admin https://localhost/api/tools?media=xml</code>
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="6"> <item xsi:type="toolsGroup" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <subgroup>Administrator Tools</subgroup> <tools> <tools name="Event Administration" id="79"/> ...removed for brevity... <tools name="UD Polling" id="111"/> </tools> </item> ...removed for brevity... <item xsi:type="toolsGroup" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <subgroup>Tools</subgroup> <tools> <tools name="Ticker" id="15"/> ...removed for brevity... <tools name="Configuration Management" id="112"/> </tools> </item> ...removed for brevity... </items></pre>
---------------	---

Users

Lists information about users. Note, from Entuity v21.0 upwards, you can also manage user lists in config sets (see 'Config Sets - Users').

URL: users

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists available users.
- POST Method - creates a new user.

GET Method detail

The list of users returned is restricted for non-administrators: only the user object corresponding to the current user is returned.

Response

Response includes a list of users. Each user has following attributes.

Name	Description
id	User id unique to the server
name	User name
serverId	Entuity Server Id on which resource resides

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/users?media=json</code>
-------	---

OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "3", "name": "admin" }], "count": 1 }</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/users?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="namedItem" name="admin" id="3" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>

POST Method details

Creates a new user.

Request

Name	Description
name	username.
password	user password.

Response

The list of users after an update, as GET method would return them.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/users?media=json -X POST -H "content-type:application/json" -d \ ' { "name" : "John", "password" : "little" }'</pre>
OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "6", "name": "John" }, { "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "3", "name": "admin" }], "count": 2 }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/users?media=xml -X POST -H "content-type:application/xml" -d \ '<userPasswordWrapper> <name>John</name> <password>little</password> </userPasswordWrapper>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="John" id="6" serverId="ee934fe2-1f4d- 4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> <item xsi:type="namedItem" name="admin" id="3" serverId="ee934fe2- 1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>

User Details

List detailed user information, modify parameters of a user, and delete a user. Note, from Entuity v21.0 upwards, you can also manage user details in config sets (see 'Config Sets - User Details').

URL: users/{userID}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - lists detailed user info.
- PUT Method - modifies parameters of a user.
- DELETE Method - deletes user.

GET Method details

Returns detailed information about the given user.

Response

User info structure with the following format:

Name	Description
id	user ID.
name	name of the user.
lockAttempts	number of failed login attempts before locking the account. Value -1 to disable.
lockDurationFailedAttempts	duration (minutes) the account is locked for after reaching maximum failed attempts.
expiryDays	Unix timestamp of the date on which the account will expire. Value -1 to disable.
timeoutMinutes	number of minutes after which session becomes inactive. Value -1 to disable.

pwChangeDays	days to password change. Value -1 to disable.
pwChangeNoticePeriod	number of days to display a warning before a password change is required. Value 0 to disable.
forcePWChange	whether the user needs to change their password.
groups	list of groups that this user is a member of.
locked	"locked" status of the user.
admin	if the user is admin.
expired	if the user account expired.
overrideGlobalUserSettings	if false, the user settings will always match the global user settings. Must be set to true to override global user settings.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/users/6?media=json</code>
OUTPUT	<pre>{ "id" : 6, "name" : "John", "lockAttempts" : 3, "lockDurationFailedAttempts" : 5, "expiryDays" : 1561814580, "timeoutMinutes" : 20, "pwChangeDays" : 90, "forcePWChange" : false, "passwordExpiryNoticePeriod" : 5, "groups" : ["All Users"], "overrideGlobalUserSettings" : false, "locked" : false, "admin" : false "expired" : false, }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/users/6?media=xml</code>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <userAccountInfo id="6" admin="false" locked="false" expired="false"> <name>John</name> <lockDurationFailedAttempts>5</lockDurationFailedAttempts> <forcePWChange>false</forcePWChange> <overrideGlobalUserSettings>true</overrideGlobalUserSettings> <passwordExpiryNoticePeriod>5</passwordExpiryNoticePeriod> <lockAttempts>3</lockAttempts> <expiryDays>1561814580</expiryDays> <timeoutMinutes>20</timeoutMinutes> <pwChangeDays>90</pwChangeDays> <groups>All Users</groups> </userAccountInfo></pre>
---------------	--

PUT Method details

Modifies parameters of a user.

Request

Name	Description
lockAttempts	number of failed login attempts before locking the account. Value -1 to disable.
lockDurationFailedAttempts	the duration (in minutes) for which the account is locked after reaching the maximum number of failed attempts.
expiryDays	Unix timestamp of the date on which the account will expire. Value -1 to disable.
timeoutMinutes	number of minutes after which session becomes inactive. Value -1 to disable.
pwChangeDays	days to password change. Value -1 to disable.
pwChangeNoticePeriod	the number of days to display a warning before a password change is required. Value 0 to disable.
forcePWChange	if the user needs to change their password.
groups	list of groups that this user is a member of.
overrideGlobalUserSettings	if false, the user settings will always match the global user settings. Must be set to true to override global user settings.

Response

Detailed information about the user after changes, as the GET method would return it.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/users/6?media=json -X PUT -H "content-type:application/json" -d \ ' { "lockAttempts" : 3, "expiryDays" : 1579800600, } '</pre>
OUTPUT	<pre>{ "id" : 6, "name" : "John", "lockAttempts" : 3, "lockDurationFailedAttempts" : 5, "expiryDays" : 1579800600, "timeoutMinutes": 20, "pwChangeDays" : 90, "forcePWChange" : false, "passwordExpiryNoticePeriod" : 5, "groups" : ["All Users"], "overrideGlobalUserSettings" : true, "locked" : false, "expired" : true "admin" : false, }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/users/6?media=xml -X PUT -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <userAccountInfo expired="false" locked="false" admin="false"> <lockAttempts>3</lockAttempts> <expiryDays>1579800600</expiryDays> </userAccountInfo>'</pre>
-------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <userAccountInfo id="6" admin="false" locked="false" expired="false" id=" 19" xmlns:ns2="https://www.entuity.com/schemas/webUI"> <name>John</name> <lockDurationFailedAttempts>5</lockDurationFailedAttempts> <forcePWChange>false</forcePWChange> <overrideGlobalUserSettings>true</overrideGlobalUserSettings> <passwordExpiryNoticePeriod>5</passwordExpiryNoticePeriod> <lockAttempts>3</lockAttempts> <expiryDays>1579800600</expiryDays> <timeoutMinutes>20</timeoutMinutes> <pwChangeDays>90</pwChangeDays> <groups>All Users</groups> </userAccountInfo></pre>
---------------	---

DELETE Method details

Deletes user account.

Response

The updated list of users, as the GET method of URL “users” would return it.

Examples

INPUT	curl -u admin:admin https://localhost/api/users/6?media=json -X DELETE
OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "3", "name": "admin" }], "count": 1 }</pre>

INPUT	curl -u admin:admin https://localhost/api/users/6?media=xml -X DELETE
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="namedItem" name="admin" id="3" serverId="ee934fe2-1f4d- 4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>
---------------	---

User's Groups

List the groups a particular user is a member of.

URL: users/{userID}/groups

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - lists the groups a particular use is a member of.

GET Method details

Returns a list of the groups that the user is a member of.

Response

Name	Description
Items	the user groups that the user is a member of.
Count	number of user groups that the user is a member of.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/users/3/groups?media=json</code>
OUTPUT	<pre>{ "items" : ["All Users", "Administrators"], "count" : 2 }</pre>

INPUT	<code>curl -u admin:admin https://localhost/users/3/groups?media=xml</code>
-------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2" xmlns:ns5="https://www.entuity.com/webrpc" xmlns:ns2=" https://www.entuity.com/schemas/webUI" xmlns:ns4="https://www.entuity.com /schemas/eventengine" xmlns:ns3="https://www.entuity.com/schemas/flow"> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">All Users</item>> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001 /XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" >Administrators</item>> </items></pre>
---------------	---

Global User Settings

List global user settings, modify global user settings, and update all existing users to those global user settings.

URL:

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - returns details about global user settings.
- PUT Method - modifies global user settings.
- POST Method - update all existing users to the global settings.

GET Method details

Returns details about global user settings.

Response

Global user setting structure with the following format:

Name	Description
lockAttempts	number of failed login attempts before locking the account. Value -1 to disable.
lockDurationFailedAttempts	duration (minutes) the account is locked for after reaching maximum failed attempts.
timeoutMinutes	number of minutes after which session becomes inactive. Value -1 to disable.
pwChangeDays	days to password change. Value -1 to disable.
pwChangeNoticePeriod	number of days to display a warning before a password change is required. Value 0 to disable.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/settings/globalUserSettings?media=json</code>
OUTPUT	<pre>{ "pwChangeDays" : 30, "pwChangeNoticePeriod" : 10, "timeoutMinutes" : 20, "lockAttempts" : 3, "lockDurationFailedAttempts" : 5, }</pre>

PUT Method details

Modifies global user settings.

Response

Name	Description
lockAttempts	number of failed login attempts before locking the account. Value -1 to disable.
lockDurationFailedAttempts	duration (minutes) the account is locked for after reaching maximum failed attempts.
timeoutMinutes	number of minutes after which session becomes inactive. Value -1 to disable.
pwChangeDays	days to password change. Value -1 to disable.
pwChangeNoticePeriod	number of days to display a warning before a password change is required. Value 0 to disable.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/settings/globalUserSettings?media=xml -X PUT -H "content-type:application/json" -d \ '{ "lockAttempts" : 5, "lockDurationFailedAttempts" : 10 }'</pre>
--------------	---

OUTPUT	<pre>{ "pwChangeDays" : 30, "pwChangeNoticePeriod" : 10, "timeoutMinutes" : 20, "lockAttempts" : 5, "lockDurationFailedAttempts" : 10, }</pre>
---------------	--

POST Method details

Updates existing users to the global user settings.

Response

Name	Description
message	if the user settings reset for all existing users was successful. User-specific settings are not affected.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/settings/globalUserSettings/resetAllUsersSettings -X POST -H "content-type:application/json"</pre>
OUTPUT	<pre>{ "message" : "Success. All existing users settings now match the global user settings. User-specific settings are not affected." }</pre>

Global Password Complexity Settings

List global password complexity settings, and modify global password complexity settings.

URL:

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - returns global password complexity settings.
- PUT Method - modifies global password complexity settings.

GET Method details

Returns details about global password complexity settings.

Response

Global password complexity settings structure with the following format:

Name	Description
pwdMinLength	minimum length of password.
pwdHistoryLimit	prevent recent previous passwords from being reused (limit 10).
pwdReqUpperCase	specifies if password needs uppercase.
pwdReqLowerCase	specifies if password needs lowercase.
pwdReqNumeric	specifies if password needs numeric characters.
pwdReqSpecial	specifies if password needs special characters.
pwdReqOTP	when a user is created, the password is automatically expired.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/settings/globalPasswordComplexitySettings</code>
OUTPUT	<pre>{ "pwdHistoryLimit" : 1, "pwdMinLength" : 4, "pwdReqNumeric" : false, "pwdReqLowerCase" : false, "pwdReqUpperCase" : false, "pwdReqSpecial" : false, "pwdReqOTP" : false }</pre>

PUT Method details

Modifies global user settings.

Response

Global password complexity settings structure with the following format:

Name	Description
pwdMinLength	minimum length of password.
pwdHistoryLimit	prevent recent previous passwords from being reused (limit 10).
pwdReqUpperCase	specifies if password needs uppercase.
pwdReqLowerCase	specifies if password needs lowercase.
pwdReqNumeric	specifies if password needs numeric characters.
pwdReqSpecial	specifies if password needs special characters.
pwdReqOTP	when a user is created, the password is automatically expired.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/settings /globalPasswordComplexitySettings -X PUT -H "content-type :application /json" -d \ ' { "pwdHistoryLimit" : 4, "pwdMinLength" : 8, "pwdReqNumeric" : true, "pwdReqLowerCase" : true, "pwdReqUpperCase" : true, "pwdReqSpecial" : true, "pwdReqOTP" : true } '</pre>
OUTPUT	<pre>{ "pwdHistoryLimit" : 4, "pwdMinLength" : 8, "pwdReqNumeric" : true, "pwdReqLowerCase" : true, "pwdReqUpperCase" : true, "pwdReqSpecial" : true, "pwdReqOTP" : true }</pre>

View List

List available Views or create a new View. Note, from Entuity v21.0 upwards, you can also manage View lists in config sets (see 'Config Sets - List and Add Views').

URL: views

- [Method Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- GET Method - lists available Views.
- POST Method - create a new View.

GET Method detail

Lists available Views.

Response

Response includes a list of Views. Each View has the following attributes.

Name	Description
displayName	View name.
id	View id unique to the server.
path	View path with forward slash as a sub-view separator.
serverId	Entuity Server Id on which resource resides.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/views?media=json</code>
--------------	---

OUTPUT	<pre>{ "items" : [{ "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : "1", "displayName" : "All Objects", "path" : "All Objects" }, { "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : "2", "displayName" : "My Network (admin)", "path" : "admin::My Network" }], "count" : 2 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/views?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="viewPathItem" path="All Objects" displayName="All Objects" id="1" serverId="821c3e87-bcdf-4fef-a5e8-7d2524928d96" /> <item xsi:type="viewPathItem" path="admin::My Network" displayName=" My Network (admin)" id="2" serverId="821c3e87-bcdf-4fef-a5e8- 7d2524928d96" /> </items></pre>

POST Method detail

Creates a new View.

Request

Request is an object which may contain a subset of following properties:

Name	Description
accessGroups	array of access objects, specifying view access permissions.

baseViewAggregation	A way to aggregate base views: One of NONE (default), UNION or INTERSECTION.
baseViewPaths	array of base views
domainFilterName	name of the domain filter to use
eventFilterName	name of the event filter to use
incidentFilterName	name of the incident filter to use
name	name of the View being created.
owner	username of the user who will be an owner of the View. This defaults to the user making a call. Only administrators may specify a user other than themselves.
parentViewPath	forward-slash separated path of the parent view. Leave out to create a top-level view.
location	location of the View, which is used by the Map dashlet in Geographical Mode.

Response

The updated list of Views, as when the GET response is used.

Examples

You can create different Views using the following commands. The following are examples for the createView.json file:

To create a top-level View with most information set to default:

INPUT	<pre>curl -u admin:admin -H "content-type:application/json" https://localhost /api/views -X POST -d \ { "name" : "Simple View" }</pre>
--------------	--

OUTPUT	<pre> { "items" : [{ "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "1", "displayName" : "My Network/All Objects", "path" : "admin::My Network/All Objects" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "2", "displayName" : "My Network", "path" : "admin::My Network" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "4", "displayName" : "My Network/Simple View", "path" : "admin::My Network/Simple View" }], "count" : 3 }</pre>
---------------	--

To create a top-level View with filter information provided:

INPUT	<pre> curl -u admin:admin -H "content-type:application/json" https://localhost /api/views -X POST -d \ ' { "name" : "MyView", "domainFilterName" : "All Objects", "eventFilterName" : "All Events", "incidentFilterName" : "All Incidents" }'</pre>
--------------	---

OUTPUT	<pre>{ "items": [{ "serverId": "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id": "1", "displayName": "My Network/All Objects", "path": "admin::My Network/All Objects", }, { "serverId": "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id": "2", "displayName": "My Network", "path": "admin::My Network" }, { "serverId": "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id": "4", "displayName": "My Network/Simple View", "path": "admin::My Network/Simple View" }, { "serverId": "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id": "6", "displayName": "My Network/MyView", "path": "admin::My Network/MyView" }], "count": 4 }</pre>
---------------	---

To create a Subview:

INPUT	<pre>curl -u admin:admin https://localhost/api/views -X POST -H "content-type: application/json" -d \ '{ "name": "SubView", "parentViewPath": "MyView" }'</pre>
--------------	---

```

OUTPUT {
  "items" : [ {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "1",
    "displayName" : "My Network/All Objects",
    "path" : "admin::My Network/All Objects"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "2",
    "displayName" : "My Network",
    "path" : "admin::My Network"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "4",
    "displayName" : "My Network/Simple View",
    "path" : "admin::My Network/Simple View"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "6",
    "displayName" : "My Network/MyView",
    "path" : "admin::My Network/MyView"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "7",
    "displayName" : "My Network/MyView/SubView",
    "path" : "admin::My Network/MyView/SubView"
  }, {
    "count" : 5
  }
}

```

To create a View that unions contents from the base Views:

INPUT	<pre>curl -u admin:admin https://localhost/api/views -X POST -H "content-type: application/json" -d \ { "name" : "Union View", "baseViewAggregation" : "UNION", "baseViewPaths" : ["MyView", "My Top View/Second-Level View/SubView"] }</pre>
-------	---

```
OUTPUT {
  "items" : [ {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "1",
    "displayName" : "My Network/All Objects",
    "path" : "admin::My Network/All Objects"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "2",
    "displayName" : "My Network",
    "path" : "admin::My Network"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "4",
    "displayName" : "My Network/Simple View",
    "path" : "admin::My Network/Simple View"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "6",
    "displayName" : "My Network/MyView",
    "path" : "admin::My Network/MyView"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "7",
    "displayName" : "My Network/MyView/SubView",
    "path" : "admin::My Network/MyView/SubView"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "8",
    "displayName" : "My Network/Union View",
    "path" : "admin::My Network/Union View"
  }, {
    "count" : 6
  }
}
```

To create a View that intersects contents from the base Views:

INPUT	<pre>curl -u admin:admin https://localhost/api/views -X POST -H "content-type: application/json" -d \ '{ "name" : "Intersection View", "baseViewAggregation" : "INTERSECTION", "baseViewPaths" : ["MyView", "MyView/SubView"] }'</pre>
OUTPUT	<pre>{ "items" : [{ "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "1", "displayName" : "My Network/All Objects", "path" : "admin::My Network/All Objects" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "2", "displayName" : "My Network", "path" : "admin::My Network" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "4", "displayName" : "My Network/Simple View", "path" : "admin::My Network/Simple View" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "6", "displayName" : "My Network/MyView", "path" : "admin::My Network/MyView" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "7", "displayName" : "My Network/MyView/SubView", "path" : "admin::My Network/MyView/SubView" }]</pre>

```

    }, {

      "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
      "id" : "8",
      "displayName" : "My Network/Union View",
      "path" : "admin::My Network/Union View"
    }, {

      "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
      "id" : "9",
      "displayName" : "My Network/Intersection View",
      "path" : "admin::My Network/Intersection View"
    }, {
      "count" : 7
    }
  }

```

To create a View with a specific owner and access rights:

INPUT	<pre> curl -u admin:admin https://localhost/api/views -X POST -H "content-type: application/json" -d \ ' { "name" : "John's View", "owner" : "John", "accessGroups" : [{ "userGroupName" : "Support", "editable" : "true" }, { "userGroupName" : "All Users", "editable" : "false" }] } '</pre>
OUTPUT	<pre> { "items" : [, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "1", "displayName" : "My Network/All Objects", </pre>

```
"path" : "admin::My Network/All Objects"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "2",
  "displayName" : "My Network",
  "path" : "admin::My Network"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "4",
  "displayName" : "My Network/Simple View",
  "path" : "admin::My Network/Simple View"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "6",
  "displayName" : "My Network/MyView",
  "path" : "admin::My Network/MyView"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "7",
  "displayName" : "My Network/MyView/SubView",
  "path" : "admin::My Network/MyView/SubView"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "8",
  "displayName" : "My Network/Union View",
  "path" : "admin::My Network/Union View"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "9",
  "displayName" : "My Network/Intersection View",
  "path" : "admin::My Network/Intersection View"
}, {
  "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
  "id" : "11",
  "displayName" : "My Network/Johns View",
  "path" : "admin::My Network/Johns View"
```

```

    }, {
      "count" : 8
    }
  }

```

To create a top-level View with a specified location:

INPUT	<pre> curl -u admin:admin https://localhost/api/views -X POST -H "content-type:application/json" -d \ ' { "name" : "London Branch", "owner" : "admin", "accessGroups" : [{ "userGroupName" : "All Users", "editable" : "true" }], "location" : "london, uk" }' </pre>
OUTPUT	<pre> { "items" : [, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "1", "displayName" : "My Network/All Objects", "path" : "admin::My Network/All Objects" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "2", "displayName" : "My Network", "path" : "admin::My Network" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "4", "displayName" : "My Network/Simple View", "path" : "admin::My Network/Simple View" }, { "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0", "id" : "6", "displayName" : "My Network/MyView", </pre>

```

    "path" : "admin::My Network/MyView"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "7",
    "displayName" : "My Network/MyView/SubView",
    "path" : "admin::My Network/MyView/SubView"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "8",
    "displayName" : "My Network/Union View",
    "path" : "admin::My Network/Union View"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "9",
    "displayName" : "My Network/Intersection View",
    "path" : "admin::My Network/Intersection View"
  }, {
    "serverId" : "305df7e4-408d-4db5-a05e-b2d847ca78f0",
    "id" : "11",
    "displayName" : "My Network/Johns View",
    "path" : "admin::My Network/Johns View"
  }, {
    "serverId": "293bd62c-413f-4f79-a529-0a0c836bec84",
    "id": "12",
    "displayName": "My Network/London Branch",
    "path": "admin::My Network/London Branch"
  }],
  "count" : 9
}

```

View Details

Inspect, update or delete a View. Note, from Entuity v21.0 upwards, you can also manage View details in config sets (see 'Config Sets - Update, Rename and Delete Views').

URL: `views/{id}`

- [Method Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- GET Method - inspect a View.
- PUT Method - update a View.
- DELETE Method - delete a View.

GET Method detail

Response

Response is the entity with the following attributes:

Name	Description
accessGroups	group access permissions.
baseViewAggregation	one of NONE, UNION, INTERSECTION.
baseViewPaths	array of base View paths.
displayName	View name.
domainFilterName	domain filter name.
eventFilterName	event filter name.
id	View id unique to the server.

implicitAccessGroups	groups having implicit access by virtue of inheriting access through other group permissions, such as having access to the parent View. This may be indirect, e.g. access to the grandparent or great-grandparent.
implicitAccessUsers	users having implicit access by virtue of inheriting access through other user permissions, such as having access to the parent View. This may be indirect, e.g. access to the grandparent or great-grandparent.
incidentFilterName	incident filter name.
manuallyPopulated	if contents of the View are populated manually (true) or automatically (false).
owner	user owning a View.
path	View path.
serverId	Entuity Server Id on which resource resides.
location	geographical location of the View, used by the map dashlet in Geographical Mode. If the string is left empty, the location value will be removed from the View.
lat	latitude of the location. If latitude is specified, then a location must be specified.
lng	longitude of the location. If longitude is specified, then a location must be specified.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/views/5?media=json</code>
--------------	---

OUTPUT	<pre>{ "serverId" : "5bc28880-465b-4228-9f0c-45ab88487f83", "id" : "5", "path" : "admin::My Network/Servers", "displayName" : "My Network/Servers", "baseViewAggregation" : "NONE", "baseViewPaths" : [], "domainFilterName" : "All Objects", "manuallyPopulated" : true, "eventFilterName" : "All Events", "incidentFilterName" : "All Incidents", "owner" : "admin", "accessGroups" : [{ "userGroupName" : "Administrators", "editable" : true }], "implicitAccessGroups" : [], "implicitAccessUsers" : [] "location" : "", "lat" : 0.0, "lng" : 0.0 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/views/48?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <viewPathDetails owner="admin" incidentFilterName="All Incidents" eventFilterName="All Events" manuallyPopulated="true" domainFilterName=" All Objects" baseViewAggregation="NONE" displayName="Simple View" path=" Simple View" id="48" serverId="821c3e87-bcdf-4fef-a5e8-7d2524928d96"> <accessGroup editable="true" userGroupName="Administrators"/> </viewPathDetails></pre>

PUT Method detail

Update existing View.

Request

Request has the same structure as the POST request of views resource for adding a new view, except the following property must be absent: **parentViewPath**. You need to specify only properties you want to be changed.

Response

Response has the same structure as the GET response: shows View details after the update.

Examples

```
curl -u admin:admin -H "content-type:application/json" -X PUT --data @updateView.json https://localhost/api/views/48
```

INPUT	<pre>curl -u admin:admin https://localhost/api/views/5 -X PUT -H "content-type:application/json" -d \ ' { "owner" : "user" } '</pre>
OUTPUT	<pre>{ "serverId" : "5bc28880-465b-4228-9f0c-45ab88487f83", "id" : "5", "path" : "admin::My Network/Servers", "displayName" : "My Network/Servers", "baseViewAggregation" : "NONE", "baseViewPaths" : [], "domainFilterName" : "All Objects", "manuallyPopulated" : true, "eventFilterName" : "All Events", "incidentFilterName" : "All Incidents", "owner" : "user", "accessGroups" : [{ "userGroupName" : "Administrators", "editable" : true }], "implicitAccessGroups" : [], "implicitAccessUsers" : [] "lat" : 0.0, "lng" : 0.0 }</pre>

DELETE Method detail

Delete existing View.

Request

No request expected.

Response

Empty response.

Examples

Delete View with id 48.

INPUT	<code>curl -u admin:admin https://localhost/api/views/48 -X DELETE</code>
OUTPUT	<code>[]</code>

View Objects

Inspect, add or delete View objects.

URL: `views/{id}/objects`

- [Method Summary](#)
- [GET Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- GET Method - list View objects.
- PUT Method - add objects to a View.
- DELETE Method - remove objects from a View.

GET Method detail

Method returns objects contained in a View.

Request

The following query parameter can be specified:

Name	Description
indirect	set to a special value "include" to return objects from Subviews as well. By default, objects from Subviews are not included.
includedTypes	specify the types of object to include in the returned list of StormWorks type names, e.g. Service, VPNDevice. Can be used in conjunction with <code>excludedTypes</code> below.
excludedTypes	specify the types of object to exclude from the returned list of StormWorks type names. Can be used in conjunction with <code>includedTypes</code> above.

Response

List of items. Each item has the following properties:

Name	Description
displayName	name of the object.
id	object id unique to the server.
serverId	Entuity Server Id on which resource resides.
typeDisplayName	user-friendly name of the object type.
typeName	name of the object type.

Examples

INPUT	curl -u admin:admin https://localhost/api/views/49/objects?media=json
OUTPUT	<pre>{ "items" : [{ "serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96", "id" : 769, "typeName" : "ManagedHost", "typeDisplayName" : "Managed Host", "displayName" : "localhost" }], "count" : 1 }</pre>

INPUT	curl -u admin:admin https://localhost/api/views/49/objects?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="viewContentItem" displayName="localhost" typeDisplayName="Managed Host" typeName="ManagedHost" id="769" serverId="821c3e87-bcdf-4fef-a5e8- 7d2524928d96" /> </items></pre>

- To return **all** devices:

```
curl -u admin:admin https://localhost/api/views/49/objects?media=json&includedTypes=BasicDevice
```

- To return **only** services and VPN devices:

```
curl -u admin:admin https://localhost/api/views/49/objects?media=json&includedTypes=Service,VPNDevice
```

- To return all devices **except** VPN devices:

```
curl -u admin:admin https://localhost/api/views/49/objects?media=json&includedTypes=BasicDevice&excludedTypes=VPNDevice
```

PUT Method detail

Request

Following query parameters can be specified

Name	Description
id	item id. This parameter can appear multiple times.

Response

Response is the same as from a GET method after objects have been added.

Examples

Add two items (2024 and 3279) to a View:

```
curl -u admin:admin https://localhost/api/views/49/objects?media=json&id=2024&id=3279 -X PUT
```

DELETE Method detail

Request

The following query parameters can be specified:

Name	Description
id	object id. This parameter can appear multiple times.
include	Can have a special value "all" to empty a View completely. If specified any 'id' parameters are ignored.

Response

The updated list of View objects, as would be received from the GET method.

Examples

Remove two objects (2024 and 3279) from a View:

```
curl -u admin:admin https://localhost/api/views/49/objects?  
media=json&id=2024&id=3279 -X DELETE
```

Remove all items from a View:

```
curl -u admin:admin https://localhost/api/views/49/objects?media=json&include=all -  
X DELETE
```

Object Attributes

Lists attributes of an object.

URL: `objects/{swID}/attributes?includeDetails=true&name=name1&name=name2`

- [Methods summary](#)
- [GET Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - lists attributes of the object.

GET Method details

Lists the attributes of the object.

Request

The request may contain a list of attribute names to retrieve in the format “name=value”. By default, all attributes are returned.

It is also possible to supply “includeDetails” flag. It is assumed to be true if this value is set to one of: “true”, “yes”, “t”, “y” or “1”. If it is set to any other value, it is false. If a value is absent, then it is automatically set to true if there is at least one “name=value” entry, otherwise it is false..

The “includeDetails” controls the level of details returned by this method. If false, only the attribute names will be returned. If true, attribute names together with their values will be returned.

Response

An array of entries, as described below.

If “includeDetails” is false, every entry has the following format:

Name	Description
Name	name of the attribute.

If “includeDetails” is true, every entry has the following format:

Name	Description
------	-------------

name	value of "name" attribute of the device.
displayName	value of "displayName" attribute of the device
userEditable	if the attribute can be modified by the user.
userOverriden	if the attribute is set to polled or user-defined value.
values	list of values for the attribute.

Examples

INPUT	curl -u admin:admin https://localhost/api/objects/611/attributes?media=json&includeDetails=y
OUTPUT	<pre>{ "items" : [{ "name" : "devType", "displayName" : "Device Type", "userEditable" : false, "userOverriden" : false, "values" : ["148"] }, { ...removed for brevity... }, { "name" : "category", "displayName" : "category", "userEditable" : false, "userOverriden" : false, "values" : ["device"] }], "count" : 30 }</pre>

INPUT	curl -u admin:admin https://localhost/api/objects/611/attributes?media=xml&includeDetails=y
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="30"> <item xsi:type="attribute" userOverriden="false" userEditable="false" displayName="Device Type" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"> <name>devType</name> <value>148</value> </item> ...removed for brevity... <item xsi:type="attribute" userOverriden="false" userEditable="false" displayName="category" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"> <name>category</name> <value>device</value> </item> </items></pre>
---------------	---

Object Attribute Details

List values of an object attribute, and modify an object attribute.

URL: objects/{swID}/attributes/{attributeName}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - lists values of an attribute of the object.
- PUT Method - modifies an attribute of the object.

GET Method details

Lists the values of an attribute on this object.

Response

The list of attributes, with entries according to the following format:

Name	Description
name	value of "name" attribute of the device.
displayName	value of "displayName" attribute of the device.
userEditable	if the attribute can be user-modified.
userOverriden	if the attribute is set to polled or a user-defined value.
values	list of values for the attribute.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/objects/611/attributes/typeName?media=json</code>
-------	---

OUTPUT	<pre>{ "name" : "typeName", "displayName" : "typeName", "userEditable" : false, "userOverriden" : false, "values" : ["SwitchDevice"] }</pre>
---------------	--

INPUT	<code>curl -u admin:admin https://localhost/api/objects/611/attributes/typeName?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <objectAttribute userOverriden="false" userEditable="false" displayName=" typeName"> <name>typeName</name> <value>SwitchDevice</value> </objectAttribute></pre>

PUT Method details

Modifies an attribute of the object.

Request

An attribute definition:

Name	Description
name	name of attribute to be changed.
userOverriden	if the attribute has user-supplied or polled value. The fields "userOverriden" and "values" are mutually exclusive, meaning that "values" are only permitted if the "userOverriden" is false.
values	list of values for the attribute.

Response

An updated attribute:

Name	Description
------	-------------

name	value of "name" attribute of the device.
displayName	value of "displayName" attribute of the device.
userEditable	if the attribute can be modified by the user.
userOverriden	if you want to override the value, you must set "userOverriden" to "true".
values	list of values for this attribute.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/611/attributes /TransferServer?media=json -X PUT -H "content-type:application/json" -d \ '{ "name" : "TransferServer", "displayName" : "Configuration Transfer Server", "userOverriden" : true, "values" : ["10.44.2.101"] }'</pre>
OUTPUT	<pre>{ "name" : "TransferServer", "displayName" : "Configuration Transfer Server", "userEditable" : true, "userOverriden" : true, "values" : ["10.44.2.101"] }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/611/attributes /TransferServer?media=xml -X PUT -H "content-type:application/xml" -d \ '<objectAttribute userOverriden="true" userEditable="true" displayName=" Configuration Transfer Server"> <name>TransferServer</name> <value>10.44.2.101</value> </objectAttribute>'</pre>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <objectAttribute userOverriden="true" userEditable="true" displayName=" Configuration Transfer Server"> <name>TransferServer</name> <value>10.44.2.101</value> </objectAttribute></pre>
---------------	--

Object Ports

Lists the ports on an object.

URL: `objects/{swID}/ports?includeVirtual=BOOL&includeUnmanaged=BOOL`

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - lists ports on the object.

GET Method details

This method accepts one of two boolean parameters:

- one that equals "yes", "true", "y", "t" or "1". This is treated as 'true'.
- one that equals any other value, or the absence of a value. This is treated as 'false'.

`includeVirtual` – controls whether virtual ports should be included in results.

`includeUnmanaged` – controls whether data for unmanaged ports should be included.

Response

The list of attributes, with entries according to the following format:

Name	Description
objectId	StormWorks ID of the associated object.
typeName	type of the associated object.
displayName	display name.
displayType	display type.
hasServiceStatus	if object has a service status.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/objects/655/ports?includeVirtual=yes&includeUnmanaged=yes&media=json</code>
-------	---

```

OUTPUT {
  "items": [ {
    "portAttributes": {
      "compId": {
        "ids": [ 1, 2, 1, 0 ],
        "type": 1,
        "invalid": false,
        "port": true,
        "networkPath": false,
        "device": false,
        "root": false,
        "dsObject": false,
        "view": false
      },
      "connectedHostIps": "",
      "connectedHostMacs": "",
      "connectedHostNames": "",
      "connectedHosts": "",
      "displayName": " [ Gi0/0 ] to 10.44 lan",
      "ifDescr": " [ Gi0/0 ] to 10.44 lan",
      "ifIndex": "1",
      "ifType": "Ethernet",
      "ipAddresses": "10.44.1.59",
      "objectId": 691,
      "portAdminStatus": "up",
      "portAlias": "to 10.44 lan",
      "portDescr": "GigabitEthernet0/0",
      "portDuplex": "Full Duplex",
      "portInSpeed": 100000000,
      "portMac": "00:07:0e:34:f4:00",
      "portOperationalStatus": "up",
      "portOutSpeed": 100000000,
      "portShortDescr": "[ Gi0/0 ]",
      "portSpare": "No",
      "portVipStatus": "Router",
      "portVirtualIndicator": "Physical",
      "statusEventsEnabled": 1,
    }
  ]
}

```

```

    "statusFasterPolling": 0,
    "timeOfLastChange": 1458210600,
    "topoNodeState": 0,
    "typeDisplayName": "Router Device",
    "typeName": "portEx",
    "utilFasterPolling": 0,
    "vlans": ""
  }
}, {
...removed for brevity...
}, {
  "portAttributes": {
    "compId": {
      "ids": [
        1,
        2,
        12,
        0
      ],
      "type": 1,
      "invalid": false,
      "port": true,
      "networkPath": false,
      "device": false,
      "root": false,
      "dsObject": false,
      "view": false
    },
    "connectedHostIps": "",
    "connectedHostMacs": "",
    "connectedHostNames": "",
    "connectedHosts": "",
    "displayName": " [ Se0/1/0 ] Serial0/1/0-mpls layer",
    ...removed for brevity...
    "typeDisplayName": "Router Device",
    "typeName": "portEx",

```

```

    "utilFasterPolling": 0,
    "vlans": ""
  }
} ],
"count": 8
}

```

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/655/ports?includeVirtual=yes&includeUnmanaged=yes&media=xml</pre>
OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="8"> <item xsi:type="devicePortInfo" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"> <portAttributes> <compId> <ids> <ids>1</ids> <ids>2</ids> <ids>1</ids> <ids>0</ids> </ids> </compId> <connectedHostIps></connectedHostIps> <connectedHostMacs></connectedHostMacs> <connectedHostNames></connectedHostNames> <connectedHosts></connectedHosts> <displayName> [Gi0/0] to 10.44 lan</displayName> <ifDescr> [Gi0/0] to 10.44 lan</ifDescr> <ifIndex>1</ifIndex> <ifType>Ethernet</ifType> <ipAddresses>10.44.1.59</ipAddresses> <objectId>691</objectId> <portAdminStatus>up</portAdminStatus> <portAlias>to 10.44 lan</portAlias> <portDescr>GigabitEthernet0/0</portDescr> </item> </items> </pre> </pre>

```

    <portDuplex>Full Duplex</portDuplex>
    <portInSpeed>100000000</portInSpeed>
    <portMac>00:07:0e:34:f4:00</portMac>
    <portOperationalStatus>up</portOperationalStatus>
    <portOutSpeed>100000000</portOutSpeed>
    <portShortDescr>[ Gi0/0 ]</portShortDescr>
    <portSpare>No</portSpare>
    <portVipStatus>Router</portVipStatus>
    <portVirtualIndicator>Physical</portVirtualIndicator>
    <statusEventsEnabled>1</statusEventsEnabled>
    <statusFasterPolling>0</statusFasterPolling>
    <timeOfLastChange>1458210600</timeOfLastChange>
    <topoNodeState>0</topoNodeState>
    <typeDisplayName>Router Device</typeDisplayName>
    <typeName>portEx</typeName>
    <utilFasterPolling>0</utilFasterPolling>
    <vlans></vlans>
  </portAttributes>
</item>
...removed for brevity...
  <item xsi:type="devicePortInfo" xmlns:xsi="https://www.w3.org/2001
/XMLSchema-instance">
    <portAttributes>
      <compId>
        <ids>
          <ids>1</ids>
          <ids>2</ids>
          <ids>12</ids>
          <ids>0</ids>
        </ids>
      </compId>
      <connectedHostIps></connectedHostIps>
      <connectedHostMacs></connectedHostMacs>
      <connectedHostNames></connectedHostNames>
      <connectedHosts></connectedHosts>
      <displayName> [ Se0/1/0 ] Serial0/1/0-mpls layer</displayName>
    </portAttributes>
    ...removed for brevity...
  </item>

```

```
<typeDisplayName>Router Device</typeDisplayName>
<typeName>portEx</typeName>
<utilFasterPolling>0</utilFasterPolling>
<vlans></vlans>
</portAttributes>
</item>
</items>
```

Object Associations

Lists an object's associations.

URL: `objects/{swID}/associations?showEmpty=BOOL`

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- Lists associations of the object.

GET Method details

Lists associations of the object.

This method accepts a “showEmpty” parameter, which controls whether empty associations should be listed. This method accepts one of two boolean parameters:

- one that equals “yes”, “true”, “y”, “t” or “1”. This is treated as ‘true’.
- one that equals any other value, or the absence of a value. This is treated as ‘false’.

Response

The list of associations.

Name	Description
Name	Association name

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/objects/655/associations?showEmpty=yes&media=json</code>
-------	--

OUTPUT	<pre>{ "items": ["AdaptorUnits", "Annotation", "ChassisList", "ComputeBlades", "DeviceMIBPolls", "FabricExtenders", "FanModules", "Fans", "HostEthernets", "IPSLABaseCreators", "IPSLABasePollers", "IpAddresses", "LocalDisks", "...removed for brevity...", "uDComponents19", "uDComponents20"], "count": 60 }</pre>
---------------	--

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/655/associations? showEmpty=yes&media=xml</pre>
--------------	--

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="60"> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">AdaptorUnits</item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Annotation</item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">ChassisList</item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">ComputeBlades< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">DeviceMIBPolls< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">FabricExtenders< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">FanModules</item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">Fans</item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">HostEthernets< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">IPSLABaseCreators< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">IPSLABasePollers< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">IpAddresses</item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">LocalDisks</item> ...removed for brevity... <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">uDComponents19< /item> <item xsi:type="xs:string" xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">uDComponents20< /item> </items> </pre>
---------------	---

Association Details

Display details of an association.

URL: `objects/{swID}/associations/{associationName}`

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - displays details of an association.

GET Method details

Displays details of an association. Association names are not case sensitive.

Response

The list of items, with entries according to the following format:

Name	Description
objectId	StormWorks ID of the associated object.
typeName	type of the associated object.
displayName	display name.
displayType	display type.
hasServiceStatus	if object has service status.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/objects/655/associations/fans?media=json</code>
-------	---

OUTPUT	<pre>{ "items": [{ "objectId": 656, "typeName": "Fan", "displayName": "Fan 1", "displayType": "", "hasServiceStatus": true }, { "objectId": 657, "typeName": "Fan", "displayName": "Fan 2", "displayType": "", "hasServiceStatus": true }, { "objectId": 658, "typeName": "Fan", "displayName": "Fan 3", "displayType": "", "hasServiceStatus": true }], "count": 3 }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/655/associations/fans? media=xml</pre>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="3"> <item xsi:type="associationInfo" hasServiceStatus="true" displayType="" displayName="Fan 1" typeName="Fan" objectId="656" xmlns:xsi="https://www. w3.org/2001/XMLSchema-instance"/> <item xsi:type="associationInfo" hasServiceStatus="true" displayType="" displayName="Fan 2" typeName="Fan" objectId="657" xmlns:xsi="https://www. w3.org/2001/XMLSchema-instance"/> <item xsi:type="associationInfo" hasServiceStatus="true" displayType="" displayName="Fan 3" typeName="Fan" objectId="658" xmlns:xsi="https://www. w3.org/2001/XMLSchema-instance"/> </items></pre>
---------------	---

Configuration Management

List the configuration management settings of an object, and modify those settings.

URL: objects/{swID}/configManagement

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - lists configuration management settings of the object.
- PUT Method - modifies configuration management attributes of the object.

GET Method details

Lists configuration management settings of the object.

Response

The list of configuration management settings of the object:

- NumberOfConfigsToArchive
- configMonitorRetrievalScript
- configMonitorPolicyRules
- ConfigRetrievalEnabled
- configMonitorTransferMethod
- SNMPChangeDetectionEnabled
- configMonitorExcludedDifference
- cliMethod
- cliUsername.

Every entry will follow the description below:

Name	Description
name	value of "name" attribute of the device
displayName	value of "displayName" attribute of the device.
userEditable	if the attribute can be modified by the user.

userOverriden	if the attribute is set to polled or a user-defined value.
values	list of values for this attribute.

Examples

INPUT	curl -u admin:admin https://localhost/api/objects/605/ports?includeVirtual=y&includeUnmanaged=y&media=json
OUTPUT	<pre>{ "items" : [{ "portAttributes" : { "compId" : { "ids" : [1, 2, 1, 0], "type" : 1, "invalid" : false, "root" : false, "networkPath" : false, "device" : false, "port" : true, "view" : false, "dsObject" : false }, "connectedHostIps" : "", "connectedHostMacs" : "", "connectedHostNames" : "", "connectedHosts" : "", "displayName" : " [V11] Vlan1", "ifDescr" : " [V11] Vlan1", "ifIndex" : "1", "ifType" : "Prop. Virtual/Internal", "ipAddresses" : "10.44.1.41", "objectId" : 723, "portAdminStatus" : "up", "portAlias" : "", "portDescr" : "Vlan1", "portDuplex" : "Unknown", "portInSpeed" : 0, "portMac" : "00:19:06:d2:1e:c0",</pre>

```

    "portOperationalStatus" : "up",
    "portOutSpeed" : 0,
    "portShortDescr" : "[ V11 ]",
    "portSpare" : "No",
    "portVipStatus" : " ",
    "portVirtualIndicator" : "Virtual",
    "statusEventsEnabled" : 0,
    "statusFasterPolling" : 0,
    "timeOfLastChange" : null,
    "topoNodeState" : 292,
    "typeDisplayName" : "Switch Device",
    "typeName" : "portEx",
    "utilFasterPolling" : 0,
    "vlans" : ""
  }
}, {
  ...removed for brevity...
}, {
  "portAttributes" : {
    "compId" : {
      "ids" : [ 1, 2, 6, 0 ],
      "type" : 1,
      "invalid" : false,
      "root" : false,
      "networkPath" : false,
      "device" : false,
      "port" : true,
      "view" : false,
      "dsObject" : false
    },
    "connectedHostIps" : "",
    "connectedHostMacs" : "",
    "connectedHostNames" : "",
    "connectedHosts" : "",
    "displayName" : " [ Fa0/4 ] FastEthernet0/4",
    "ifDescr" : " [ Fa0/4 ] FastEthernet0/4",

```

```
...removed for brevity...
}
}, {
...removed for brevity...
}, {
  "portAttributes" : {
    "compId" : {
      "ids" : [ 1, 2, 18, 0 ],
      "type" : 1,
      "invalid" : false,
      "root" : false,
      "networkPath" : false,
      "device" : false,
      "port" : true,
      "view" : false,
      "dsObject" : false
    },
    "displayName" : " [ Fa0/16 ] FastEthernet0/16",
    "objectId" : 18,
    "typeName" : "UnmanagedPort"
  }
} ],
"count" : 29
}
```

PUT Method details

Modifies a single configuration management attribute of the device.

Request

An attribute definition:

Name	Description
------	-------------

Name	name of attribute that is to be modified. The only valid values are: • NumberOfConfigsToArchive • configMonitorRetrievalScript • configMonitorPolicyRules • ConfigRetrievalEnabled • configMonitorTransferMethod • SNMPChangeDetectionEnabled • configMonitorExcludedDifference • cliMethod • cliUsername.
values	list of values for this attribute

Response

An updated attribute:

“OK” if the method succeeded, an error description otherwise.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/611/configManagement? media=json -X PUT -H "content-type:application/json" -d \ '{"name" : "cliMethod", "values" : ["ssh"] }'</pre>
OUTPUT	<pre>{ "message" : "OK" }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/611/configManagement? media=xml -X PUT -H "content-type:application/xml" -d \ '<objectAttribute> <name>cliMethod</name> <value>ssh</value> </objectAttribute>'</pre>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>
---------------	---

Port Management

Set the managed/unmanaged state of a port.

URL: portManagement/{devID}/{portID}?reManage=BOOL&unManage=BOOL

- [Methods summary](#)
- [PUT Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- PUT Method - controls the managed/unmanaged state of the port.

PUT Method details

This method can accept two boolean parameters, which can be equal to either of “yes”, “true”, “y”, “t” or “1”.

If both parameters are present, “reManage” takes over the “unManage”.

If none are present, then this method reports an error.

Response

A message, describing the current state of a port.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/portManagement/2/18?unManage=y&media=json -X PUT -H "content-type:application/json"</code>
OUTPUT	<pre>{ "message" : "Successfully marking port as unmanaged" }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/portManagement/2/18?unManage=y&media=xml -X PUT -H "content-type:application/xml"</code>
-------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Successfully marking port as unmanaged</message> </statusInfo></pre>
---------------	---

Zones

Lists configured zones and creates new zones.

URL: zones

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists configured zones.
- POST Method - creates a new zone.

GET Method detail

Lists configured zones.

Response

Response includes a list of zones. Each zone has following attributes.

Name	Description
id	zone ID, unique to the server.
name	zone name.
serverId	Entuity Server ID on which resource resides.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/zones?media=json</code>
-------	---

OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "0", "name": "None" }], "count": 1 }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/zones?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="namedItem" name="None" id="0" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>

POST Method details

Creates a new zone.

Request

Name	Description
id	zone ID.
name	name of the zone.
flags	zone flags, should be set to 0.
description	description of the zone.
v4interface	IPv4 interface for the zone.
v6interface	IPv6 interface for the zone.
domainSuffix	domain suffix for the zone.

proxy	(this is unused.)
devicePrefix	specified device prefix that is included with each device name found under this zone, maximum of 5 characters.
hostFile	host file.
dnsServer	list of DNS servers.

Response

The list of zones after an update, as GET method would return them.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/zones?media=json -X POST -H "content-type:application/json" -d \ ' { "name" : "A", "flags" : 0, "description" : "", "v4Interface" : "10.44.2.103", "v6Interface" : "", "domainSuffix" : "", "proxy" : "", "devicePrefix" : "", "hostFile" : "", "dnsServers" : ["a.b.c.d"] }</pre>
--------------	---

OUTPUT	<pre>{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "1", "name": "A" }, { "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "0", "name": "None" }], "count": 2 }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin https://localhost/api/zones?media=xml -X POST -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <zoneParametersInfo> <name>A</name> <flags>0</flags> <description></description> <v4Interface>10.44.2.103</v4Interface> <v6Interface></v6Interface> <domainSuffix></domainSuffix> <proxy></proxy> <devicePrefix></devicePrefix> <hostFile></hostFile> <dnsServers> <server>a.b.c.d</server> </dnsServers> </zoneParametersInfo>'</pre>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="namedItem" name="A" id="1" serverId="ee934fe2-1f4d-4f1e- a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" /> <item xsi:type="namedItem" name="None" id="0" serverId="ee934fe2-1f4d- 4f1e-a0b5-a87b261eb30a" xmlns:xsi="https://www.w3.org/2001/XMLSchema- instance"/> </items></pre>
---------------	--

Zone Details

This resource implements operations acting on a single zone.

URL: zones/{zoneID}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Examples](#)

Methods summary

- GET Method - details information about the zone.
- PUT Method - modifies parameters of the zone.
- DELETE Method - deletes the zone.

GET Method details

Returns a detailed information about the zone.

Response

User info structure with the following format:

Name	Description
id	zone ID.
name	name of the zone.
flags	zone flags.
description	description of the zone.
v4interface	IPv4 interface for the zone.
v6interface	IPv6 interface for the zone.
domainSuffix	Domain suffix for the zone.

proxy	// TODO.
devicePrefix	// TODO.
hostFile	host file.
dnsServer	list of DNS servers.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/zones/1?media=json</code>
OUTPUT	<pre>{ "id": 1, "name": "A", "flags": 0, "description": "", "v4Interface": "10.44.2.103", "v6Interface": "", "domainSuffix": "", "proxy": "", "devicePrefix": "", "hostFile": "", "dnsServers": ["a.b.c.d"] }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/zones/1?media=xml</code>
--------------	--

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <zoneParametersInfo id="1"> <name>A</name> <flags>0</flags> <description></description> <v4Interface>10.44.2.103</v4Interface> <v6Interface></v6Interface> <domainSuffix></domainSuffix> <proxy></proxy> <devicePrefix></devicePrefix> <hostFile></hostFile> <dnsServers> <server>a.b.c.d</server> </dnsServers> </zoneParametersInfo> </pre>
---------------	---

PUT Method details

Modifies parameters of the zone.

Request

Name	Description
id	zone ID.
name	name of the zone
flags	zone flags.
description	description of the zone.
v4interface	IPv4 interface for this zone.
v6interface	IPv6 interface for this zone.
domainSuffix	domain suffix for this zone.
proxy	// TODO
devicePrefix	// TODO
hostFile	host file.
dnsServer	list of DNS servers.

Response

Detailed information about the zone after changes, as GET method would return it.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/zones/ 1?media=json -X PUT -H "content-type:application/json" -d \ ' { "name": "B", "v6Interface": "fe80:0:0:0:0:5efe:a2c:266", "dnsServers": ["t.x.y.z"] }'</pre>
OUTPUT	<pre>{ "id": 1, "name": "B", "flags": 0, "description": "", "v4Interface": "10.44.2.103", "v6Interface": "fe80:0:0:0:0:5efe:a2c:266", "domainSuffix": "", "proxy": "", "devicePrefix": "", "hostFile": "", "dnsServers": ["t.x.y.z"] }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/zones/1?media=xml-X PUT -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <zoneParametersInfo id="1"> <name>Zone B</name> <v6Interface>fe80:0:0:0:0:5efe:a2c:266</v6Interface> <dnsServers> <server>t.x.y.z</server> </dnsServers> </zoneParametersInfo>'</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <zoneParametersInfo id="1"> <name>B</name> <flags>0</flags> <description></description> <v4Interface>10.44.2.103</v4Interface> <v6Interface></v6Interface> <domainSuffix></domainSuffix> <proxy></proxy> <devicePrefix></devicePrefix> <hostFile></hostFile> <dnsServers> <server>t.x.y.z</server> </dnsServers> </zoneParametersInfo></pre>

DELETE Method details

Deletes a zone.

Response

The current list of zones, as the GET method would return it.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/zones/1?media=json -X DELETE</pre>
--------------	---

OUTPUT	<pre>{ "message" : "Zone deleted successfully" }</pre>
---------------	--

INPUT	<pre>curl -u admin:admin https://localhost/api/zones/1?media=xml -X DELETE</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Zone deleted successfully</message> </statusInfo></pre>

IP SLA Management

View information about, modify and delete an IP SLA.

URL: ipsla/{id}

- [Methods Summary](#)
- [Request and Response](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - displays information about the IP SLA.
- PUT Method - modifies a particular IP SLA creator.
- DELETE Method - deletes a particular IP SLA creator.

Request and Response

The following attributes must be present for every IP SLA type:

Name	Description
id	IP SLA object ID
name	name
index	association index, used to match IP SLA creators and pollers
tag	zone flags
lifetime	creator's lifetime
frequency	frequency of polling
timeout	polling timeout
vrfName	VRF (Virtual Routing and Forwarding) instance name
failureCause	cause of failure

changesPending	changes pending
type	creator type, one of the following: • DHCP • DNS • ECHO • ECHO_PATH • HTTP • HTTP_RAW • JITTER • VOIP • TCP • UDP

In addition, each type provides a specific set of attributes, as show:

IP SLA type	Required attributes
DHCP	targetAddress sourceAddress sourcePort
DNS	targetAddress nameserver
ECHO and ECHO_PATH	targetAddress sourceAddress sourcePort typeOfService packetSize
HTTP	targetURL proxyAddress sourceAddress sourcePort typeOfService httpVersion useCache

HTTP_RAW	targetURL proxyAddress sourceAddress sourcePort typeOfService httpVersion useCache, adminStr1 adminStr2 adminStr3 adminStr4 adminStr5
JITTER	targetAddress targetPort sourceAddress sourcePort typeOfService interval numberOfPackets packetSize
VOIP	targetAddress targetPort sourceAddress sourcePort typeOfService codecType codecInterval, codecPayload codecNumOfPkts icpifFactor

TCP	targetAddress targetPort sourceAddress sourcePort typeOfService ctrlPkts
UDP	targetAddress targetPort sourceAddress sourcePort typeOfService ctrlPkts packetSize

GET Method details

Returns detailed information about a particular IP SLA creator.

Response

Refer to Request and Response s section above.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/ipsla/772?media=json</code>
--------------	---

OUTPUT	<pre>{ "attributes" : { "frequency" : "300", "id" : "772", "index" : "2", "lifetime" : "forever", "name" : "B", "nameserver" : "127.0.0.1", "tag" : "a", "targetAddress" : "127.0.0.1", "timeout" : "1000", "type" : "DNS", "vrfName" : "a" } }</pre>
---------------	---

INPUT	curl -u admin:admin https://localhost/api/ipsla/772?media=xml
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ipslaDetails> <attributes> <id>772</id> <type>DNS</type> <index>2</index> <name>B</name> <tag>a</tag> <lifetime>forever</lifetime> <frequency>300</frequency> <timeout>1000</timeout> <vrfName>a</vrfName> <targetAddress>127.0.0.1</targetAddress> <nameserver>127.0.0.1</nameserver> </attributes> </ipslaDetails></pre>

PUT Method details

Modifies parameters of an IP SLA creator.

NOTE: Parameters “id”, “index” and “type” are not supposed to be modified and are therefore ignored.

Request

A set of attributes that define an IP SLA creator, according to the details described in Request and Response s section above.

Response

Actual IP SLA creator parameters, as described in Request and Response s section above.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/ipsla/1481?media=json -X PUT -H "content-type:application/json" -d \ '{> "frequency" : 2000,> "lifetime" : 0,> "name" : "Y",> "sourcePort" : 2666,> "tag" : "x",> "targetAddress" : "127.0.0.2",> "timeout" : 1000,> "vrfName" : "x"> }'</pre>
-------	--

OUTPUT	<pre>{ "attributes" : { "frequency" : "2000", "id" : "1481", "index" : "3", "lifetime" : "0", "name" : "Y", "sourceAddress" : "null", "sourcePort" : "2666", "tag" : "x", "targetAddress" : "127.0.0.2", "timeout" : "1000", "type" : "DHCP", "vrfName" : "x" } }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin https://localhost/api/ipsla/1481?media=xml -X PUT -H "content-type:application/xml" -d \ '<ipslaCreator> <name>Y</name> <tag>x</tag> <lifetime>0</lifetime> <frequency>2000</frequency> <timeout>1000</timeout> <vrfName>x</vrfName> <targetAddress>127.0.0.2</targetAddress> <sourcePort>2666</sourcePort> </ipslaCreator>'</pre>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ipslaDetails> <attributes> <id>1481</id> <type>DHCP</type> <index>3</index> <name>Y</name> <tag>x</tag> <lifetime>0</lifetime> <frequency>2000</frequency> <timeout>1000</timeout> <vrfName>x</vrfName> <targetAddress>127.0.0.2</targetAddress> <sourceAddress>null</sourceAddress> <sourcePort>2666</sourcePort> </attributes> </ipslaDetails></pre>
---------------	--

DELETE Method details

The list of users returned is restricted for non-administrators: only the user object corresponding to the current user is returned.

Response

"OK" message if the IP SLA was properly deleted, an error message otherwise.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/ipsla/772?media=json -X DELETE</code>
OUTPUT	<pre>{ "message" : "OK" }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/ipsla/772?media=xml -X DELETE</code>
--------------	--

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>
---------------	---

IP SLA Creators

Resource for managing IP SLA creators configured for a given device.

URL: `objects/{swId}/ipslaCreators`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - details information about IP SLA creators that are defined for the device.
- POST Method - creates an IP SLA creator.

GET Method details

Returns detailed information about IP SLA creators defined for the device.

Response

A list of IP SLA creators, with every entry conforming to the Request and Response description section of the [IP SLA Management page](#).

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/objects/600/ipslaCreators?media=json</code>
-------	---

OUTPUT	<pre>{ "items" : [{ "attributes" : { "failureCause" : "", "frequency" : "300", "id" : "772", "index" : "2", "lifetime" : "forever", "name" : "B", "nameserver" : "127.0.0.1", "tag" : "a", "targetAddress" : "127.0.0.1", "timeout" : "1000", "type" : "DNS", "vrfName" : "a" } },], "count" : 1 }</pre>
---------------	--

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/600/ipslaCreators? media=xml</pre>
--------------	---

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="1"> <item xsi:type="ipslaAttributesMap" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <attributes> <id>772</id> <type>DNS</type> <index>2</index> <name>B</name> <tag>a</tag> <lifetime>forever</lifetime> <frequency>300</frequency> <timeout>1000</timeout> <vrfName>a</vrfName> <failureCause /> <targetAddress>127.0.0.1</targetAddress> <nameserver>127.0.0.1</nameserver> </attributes> </item> </items> </pre>
---------------	--

POST Method details

Creates an IP SLA creator for a particular device.

Request

A set of attributes that define an IP SLA creator, according to the details described in the Request and Response description section of the [IP SLA Management page](#).

Response

An updated list of IP SLA creators, with every entry conforming to the Request and Response description section of the [IP SLA Management page](#).

Examples

INPUT	<pre>curl -u admin:adminhttps://localhost/api/objects/600/ipsIaCreators? media=json -X POST -H "content-type:application/json" -d \ ' { "frequency" : 500, "index" : 0, "lifetime" : 0, "name" : "X", "sourcePort" : "2555", "tag" : "x", "targetAddress" : "127.0.0.1", "timeout" : 1000, "type" : "DHCP", "vrfName" : "x" }'</pre>
-------	--

```
OUTPUT {
  "items" : [
    {
      "attributes" : {
        "frequency" : "300",
        "id" : "772",
        "index" : "2",
        "lifetime" : "forever",
        "name" : "B",
        "nameserver" : "127.0.0.1",
        "tag" : "a",
        "targetAddress" : "127.0.0.1",
        "timeout" : "1000",
        "type" : "DNS",
        "vrfName" : "a"
      }
    }, {
      "attributes" : {
        "frequency" : "500",
        "id" : "1481",
        "index" : "0",
        "lifetime" : "0",
        "name" : "X",
        "sourceAddress" : "null",
        "sourcePort" : "2555",
        "tag" : "x",
        "targetAddress" : "127.0.0.1",
        "timeout" : "1000",
        "type" : "DHCP",
        "vrfName" : "x"
      }
    }
  ],
  "count" : 2
}
```

INPUT	<pre>curl -u admin:admin https://localhost/api/objects/600/ipslaCreators? media=xml -X POST -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ipslaCreator> <type>DHCP</type> <index>0</index> <name>X</name> <tag>x</tag> <lifetime>0</lifetime> <frequency>500</frequency> <timeout>1000</timeout> <vrfName>x</vrfName> <targetAddress>127.0.0.1</targetAddress> <sourcePort>2555</sourcePort> </ipslaCreator>'</pre>
-------	--

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="ipslaAttributesMap" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <attributes> <id>772</id> <type>DNS</type> <index>2</index> <name>B</name> <tag>a</tag> <lifetime>forever</lifetime> <frequency>300</frequency> <timeout>1000</timeout> <vrfName>a</vrfName> <targetAddress>127.0.0.1</targetAddress> <nameserver>127.0.0.1</nameserver> </attributes> </item> <item xsi:type="ipslaAttributesMap" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <attributes> <id>1481</id> <type>DHCP</type> <index>3</index> <name>X</name> <tag>x</tag> <lifetime>0</lifetime> <frequency>500</frequency> <timeout>1000</timeout> <vrfName>x</vrfName> <targetAddress>127.0.0.1</targetAddress> <sourceAddress>null</sourceAddress> <sourcePort>2555</sourcePort> </attributes> </item> </items> </pre>
---------------	--

IP SLA Pollers

Displays information about IP SLA pollers present on a given device.

URL: `objects/{swId}/ipslaPollers`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - displays detailed information about the IP SLA pollers defined for the device.

GET Method details

Returns the detailed information about the IP SLA pollers defined for the device.

Response

A list of entries, each containing a complete attribute set of a particular IP SLA poller.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/objects/600/ipslaPollers?media=json</code>
-------	--

OUTPUT	<pre> { "items" : [{ "attributes" : { "IPSLAPollerIndex" : "111", "IPSLAPollerName" : "icmp-echo (111 on e2821)", "descriptiveAlias" : "", "id" : "605", "link" : null, "rttMonAdminCodecInterval" : "", "rttMonAdminCodecNumPackets" : "", "rttMonAdminCodecPayload" : "", ...removed for brevity... "rttMonScheduleAdminRttLife" : "forever", "rttMonStatisticsAdminNumHops" : "", "rttMonStatisticsAdminNumPaths" : "", "type" : "IPSLABasePoller" } }, { ...removed for brevity... }, { "attributes" : { ...removed for brevity... } }], "count" : 4 }</pre>
---------------	--

INPUT	<pre> curl -u admin:admin https://localhost/api/objects/600/ipslaPollers? media=xml</pre>
--------------	---

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="4"> <item xsi:type="ipslaAttributesMap" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <attributes> <rttMonEchoAdminTOS>160</rttMonEchoAdminTOS> <rttMonEchoAdminTargetAddress>10.2.90.90< /rttMonEchoAdminTargetAddress> <link xsi:nil="true"/> <rttMonEchoAdminString4></rttMonEchoAdminString4> <rttMonEchoAdminString3></rttMonEchoAdminString3> <rttMonEchoAdminString2></rttMonEchoAdminString2> <rttMonEchoAdminVrfName></rttMonEchoAdminVrfName> <rttMonAdminCodecNumPackets></rttMonAdminCodecNumPackets> <rttMonEchoAdminString1></rttMonEchoAdminString1> <type>IPSLABasePoller</type> ...removed for brevity... <rttMonEchoAdminHTTPVersion></rttMonEchoAdminHTTPVersion> <rttMonScheduleAdminRttLife>forever</rttMonScheduleAdminRttLife> <rttMonEchoAdminProxy></rttMonEchoAdminProxy> <rttMonCtrlAdminOwner>richard</rttMonCtrlAdminOwner> <rttMonAdminICPIFAdvFactor></rttMonAdminICPIFAdvFactor> </attributes> </item> <item xsi:type="ipslaAttributesMap" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> ...removed for brevity... </item> <item xsi:type="ipslaAttributesMap" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"> <attributes> ...removed for brevity... </attributes> </item> </items> </pre>
---------------	--

Data Access Interface

- [General description](#)
- [The format of the export template](#)
 - [Example template](#)
 - [Export template syntax](#)

General description

Data Access is intended as a powerful, highly configurable interface for accessing ENA's internal data (like attribute values, stream data etc.)

The interface achieves this by processing the XML “templates” that describe sets of data to be exported. As these templates may be user-defined, this gives the customers an unprecedented level of control over the data itself, as well as their layout and format.

Currently, due to an extended set of possibilities in comparison to JSON, only XML template format is supported.

The interface consists of two parts (each described in detail later on): the “template management” which can be used to retrieve and modify templates residing on a server, and the “data access” part, which produces the output based on the early defined template.

The format of the export template

Example template

The following example defines a template that would traverse the list of views (supplied by the user as a part of data output) and, based on the value of “selector” element, would traverse every device. For each device found, the element **DEVICE_OBJECT** will be created and populated with these element (in the order corresponding to their order in the template):

1. The element **name**, with a value of the **name** attribute of a device.
2. The attribute **id**, with a value of executing the **simple;id** statement on the current device.
3. The stream data **v_unifiedDeviceCPU** appearing as a **CPU** element, with all the underlying attributes.
4. The list of **PORT_OBJECT** elements, corresponding to the result of executing the **simple;ref.ports** statement on the current device. Each of these element would consist of:
 - a. The attribute **descr**, with the value of **simple;this.ifDescr** statement for the current port.
 - b. The element **status**, consisting of a list of samples, each containing two attributes: **adminStatus** (presented as an **adm** element) and **operStatus**, presented as an **oper** attribute.

An example XML Data Access Template

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataAccessTemplate name="Example template">
  <description>This is a test description</description>
  <object type="DeviceEx" tag="DEVICE_OBJECT">
    <selector>simple;ref.devices</selector>
    <filter></filter>
    <attribute asElement="true" expression="name"/>
    <attribute expression="simple;id" tag="id"/>
    <stream allAttributes="true" name="v_unifiedDeviceCPU" tag="CPU"/>
  <object type="PortEx" tag="PORT_OBJECT">
    <selector>simple;ref.ports</selector>
    <filter></filter>
    <attribute asElement="true" expression="simple;this.ifDescr" tag="Descr"/>
    <stream name="portStatus" tag="status">
      <attribute asElement="true" expression="adminStatus" tag="adm"/>
      <attribute expression="operStatus" tag="oper"/>
    </stream>
  </object>
</object>
</dataAccessTemplate>
```

Export template syntax

The following table describes all the XML nodes that can be a part of a Data Access Template. Unless stated otherwise, the order of elements is not important.

In the description below, some nodes have one of three symbols: ? (question mark), + (plus sign) or * (an asterisk) appended (in parenthesis) to their names.

These symbols are not part of the name itself, but serve to define how many times a node of this type may appear under the same parent:

- ? – the node is optional and unique (i.e. may appear one or zero times)
- * – the node is optional and not unique (i.e. may appear any number of times)
- + – the node is mandatory and not unique (i.e. must appear one or more times)

Nodes without any such symbol are implied to be mandatory and unique.

Node Name	Description
-----------	-------------

dataAccessTemplate	<i>XML element</i> node that is the root node of every single Data Access Template. The list of attributes: name, type(?) The list of child elements: description(?), object
name	<i>XML attribute</i> node defining the name of the template. One server can only store templates with different names. Renaming of a template is not supported.
description	<i>XML element</i> node that serves as a placeholder for user-defined comment. It can only have one text node, with its body being the comment string.
object	<i>XML element node</i> that defines a group of objects. The list of attributes: type, tag(?), allAttributes(?), asElement(?), asAttribute(?), asText(?), allowMultibyte Attributes asElement, as Attribute and asText are mutually exclusive. The list of child elements: selector, filter, object(*), stream(*) and attribute(*) Children of type object, stream and attribute can be interleaved. During the processing step, these nodes are expanded in order of their definition and then replaced with data according to their types.
attribute	<i>XML element node</i> that defines a StormWorks attribute to be exported. The list of attributes: expression, tag(?), asElement(?), asAttribute(?), asText(?), allowMultibyte Attributes asElement, as Attribute and asText are mutually exclusive. No child elements are permitted (attempt to do so, results in a processing failure).
stream	<i>XML element node</i> that defines a StormWorks stream to be exported. The list of attributes: name, tag(?), allAttributes(?), asElement(?), asAttribute(?), asText(?) Attributes asElement, as Attribute and asText are mutually exclusive. The list of attributes: attribute(?)
asElement asAttribute asText	<i>XML attribute</i> node that controls how data should be presented in a resulting XML file. Only one of these attributes can be set to "true" for any given node, and if none of them are set, they are inherited from parent. If still none of these attributes were set, the node is treated as if asText was set to "true". Example: The node with key equal to "SomeTag" and value equal to "SomeData" will be presented as: <SomeTag>SomeData</SomeTag>, if asElement is "true" SomeTag="SomeData", if asAttribute is "true" SomeData, if asText is "true"

tag	<i>XML attribute</i> defining the name of the type, under which all data related to this object will be anchored. This attribute may occur in object, stream and attribute nodes and is optional in all of them. If omitted in an object, it will be defaulted to the type attribute of this node. For stream node, it will be defaulted to the name attribute of a stream. For attribute node, it will be defaulted to its expression attribute, but <u>only if</u> the expression does not begin with “simple;” string (in which case the whole template is invalid and its processing will fail). Example: The following template fragment: <pre><object asElement="true" type="DeviceEx" tag="DEVICE"> will get mapped to: <DEVICE>data depending on the children of object </DEVICE></pre> If the template fragment were changed to: <pre><object asElement="true" type="DeviceEx" tag="DeviceParameters" > then the corresponding export fragments would have been changed to: <DeviceParameters>... </ DeviceParameters ></pre>
type	<i>XML attribute</i> node that defines the StormWorks type of object. The type of objects defines what attributes and streams may be access and exported.
selector	<i>XML element</i> node that is the StormWorks statement. This statement is evaluated to produce a list of children, each then undergoing subsequent processing according to their definition, etc.
filter	<i>XML element</i> node that is the StormWorks statement. This statement should evaluate to boolean value and can be used to restrict the list of children produced by a selector statement.

Parameters can also be defined as templateParameters that can be used in expressions in the DataAccessTemplate. Any parameters passed in the URL that match an attribute name will be accessible in the statement language embedded in the template as 'templateParameter.<parameter value>'. Please see **Data Access - Export Triggering: GET** for further help and information on this.

Data Access Templates Management – Listing and Creating

Lists Data Access templates used to access the internal DsKernel data, and creates new Data Access templates.

URL: dataAccessTemplates

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists all Data Access templates existing on the server.
- POST Method - creates a new Data Access template.

GET Method Details

Lists all Data Access templates existing on the server.

Response

Lists information about each of templates defined on a server. Each entry has the following attributes:

Name	Description
name	name of the template.
description	description of the template.
type	type of the template (either "USER" or "SYSTEM").

Example

INPUT	<code>curl -u admin:admin https://localhost/api/dataAccessTemplates?media=xml -X GET -H "content-type:application/xml"</code>
-------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="dataAccessTemplateSummary" description="Some description" type="SYSTEM" name="A" xmlns:xsi="https://www.w3.org/2001 /XMLSchema-instance"/> <item xsi:type="dataAccessTemplateSummary" description="" type="USER" name="B" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"/> </items></pre>
---------------	--

POST Method Details

Creates a new Data Access Template.

Request

An XML definition of the template, defined according to the Export template syntax section of the [Data Access Interface](#).

Response

A new template. If unsuccessful, a description of the error that occurred.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/dataAccessTemplates?media=xml - X POST -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <dataAccessTemplate type="USER" name="C"> <description>Some description</description> <object type="DeviceEx" tag="device"> <selector>simple;ref.devices</selector> <filter></filter> <attribute asElement="false" tag="name1" expression="name"/> <object type="PortEx" tag="port"> <selector>simple;ref.ports</selector> <filter></filter> <attribute asElement="false" tag="descr" expression="simple;this. ifDescr"/> </object> </object> </dataAccessTemplate>'</pre>
--------------	--

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataAccessTemplate type="USER" name="C">
  <description>Some description</description>
  <object type="DeviceEx" tag="device">
    <attribute expression="name" tag="name1"/>
    <object type="PortEx" tag="port">
      <attribute expression="simple;this.ifDescr" tag="descr"/>
      <selector>simple;ref.ports</selector>
      <filter></filter>
    </object>
    <selector>simple;ref.devices</selector>
    <filter></filter>
  </object>
</dataAccessTemplate>
```

Data Access Templates Management – Operations on a Single Template

Manages a single Data Access template entity.

URL: `dataAccessTemplates/{templateName}`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [PUT Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method Details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists the contents of the specific Data Access template identified in the request.
- PUT Method - modifies an existing Data Access template.
- DELETE Method - removes an existing Data Access template from a server.

GET Method Details

Response

Detailed information about the Data Access template, according to the Export template syntax section of the [Data Access Interface page](#).

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/dataAccessTemplates/C?media=xml -H "content-type:application/xml"</pre>
-------	--

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <dataAccessTemplate type="USER" name="C"> <description>Some description</description> <object type="DeviceEx" tag="device"> <attribute expression="name" tag="name1"/> <object type="PortEx" tag="port"> <attribute expression="simple;this.ifDescr" tag="descr"/> <selector>simple;ref.ports</selector> <filter></filter> </object> <selector>simple;ref.devices</selector> <filter></filter> </object> </dataAccessTemplate> </pre>
---------------	---

PUT Method Details

Modifies the existing Data Access Template.

Request

An XML definition of the template, defined according to a paragraph 2.38.2.2

Response

The modified Data Access Template. If unsuccessful, a description of the error.

Example

INPUT	<pre> curl -u admin:admin https://localhost/api/dataAccessTemplates/C? media=xml -X PUT -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <dataAccessTemplate type="USER" name="C"> <description>Some description</description> <object type="DeviceEx" tag="device"> <selector>simple;ref.devices</selector> <filter></filter> <attribute asElement="false" expression="name"/> <attribute asElement="false" expression="id"/> <object type="PortEx" tag="port"> <selector>simple;ref.ports</selector> <filter></filter> <attribute asElement="false" tag="descr" expression="simple;this. ifDescr"/> </object> </object> </dataAccessTemplate>' </pre>
OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <dataAccessTemplate type="USER" name="C"> <description>Some description</description> <object type="DeviceEx" tag="device"> <attribute expression="name" tag=""/> <attribute expression="id" tag=""/> <object type="PortEx" tag="port"> <attribute expression="simple;this.ifDescr" tag="descr"/> <selector>simple;ref.ports</selector> <filter></filter> </object> <selector>simple;ref.devices</selector> <filter></filter> </object> </dataAccessTemplate> </pre>

DELETE Method Details

Removes the existing Data Access Template.

Response

The template after modifications on success, the error description otherwise.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/dataAccessTemplates/C? media=xml -X DELETE -H "content-type:application/xml"</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo></pre>

Data Access – Export Triggering

Export user-defined Data Access templates.

URL: `dataAccess/{templateName}`

- [Methods Summary](#)
- [GET Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - exports user-defined Data Access templates.

GET Method details

Exports user-defined Data Access templates. Also, can accept parameters that will be defined as `templateParameters` that can be used in expressions in the `DataAccessTemplate`.

An export, generated according to the user-defined Data Access template.

Request

Name	Description
templateServer	ID of the server (see GET Method detail section of Servers page) from which to retrieve the template.
view	name of View of which objects will be processed. This argument may appear multiple times in which case any objects belonging to at least one of the Views will be processed.
startTime endTime	start and end timestamp for exporting a stream data. Negative values are not allowed and the default value for these parameters is 0. As a special case, if both startTime and endTime are 0, then all the streams being exported will contain only the last sample.
maxResults	maximum number of elements that will be exported.

position	next element that the export should start from. It is optional, in which case it defaults to 0:0. The result of each export will have the “position” element defined. This element consists of pairs of numbers that are separated from other pairs by commas, and the elements of a pair are separated by a colon, e.g: 1:2,3:4,5:6 After successful call that returns maximum number of elements (defined by maxResults parameter), the position will point to the next element to continue from. To achieve continuation, the position returned from the last call should be passed along as an argument to the next call – as long as the returned value is an “END” string, which signals that no more data are available.
-----------------	---

Response

An export, generated according to the user-defined Data Access template. If unsuccessful, description of the error will be returned.

Examples

Below are three examples showing the “continuation” mechanism, together with some explanation. Please note that normally the XML output generated by Data Access mechanism is not formatted – it has only been presented as formatted here for the purpose of readability.

The first example omits the **position** argument, which exports data from the very beginning:

INPUT	curl -u admin:admin https://localhost/api/dataAccess/C?maxResults=5&view=All%20Objects&media=xml -H "content-type:application/xml"
OUTPUT	<pre><C serverName="ENTLONDEV08" serverId="dae26869-46e3-46df-91a0-28ce61b28d05" templateServerId="dae26869-46e3-46df-91a0-28ce61b28d05" startTime="0" endTime="0" elementsProcessed="5" position="565:2,575:0" processingTime="0.001"> <device id="565" name="apiDevice"> <port id="572" descr=" [V11] Vlan1" /> <port id="573" descr=" [Fa0/1] FastEthernet0/1" /> <port id="574" descr=" [Fa0/2] FastEthernet0/2" /> <port id="575" descr=" [Fa0/3] FastEthernet0/3" /> </device> </C></pre>

The second one is an example of the call just after the first one, which picks up from where the previous call left off:

INPUT	curl -u admin:admin https://localhost/api/dataAccess/C?position=565:2,575:0&maxResults=5&view=All%20Objects&media=xml -H "content-type: application/xml"
OUTPUT	<pre><C serverName="ENTLONDEV08" serverId="dae26869-46e3-46df-91a0-28ce61b28d05" templateServerId="dae26869-46e3-46df-91a0-28ce61b28d05" startTime="0" endTime="0" elementsProcessed="5" position="565:2,578:0" processingTime="0"> <device id="565" name="apiDevice"> <port id="576" descr=" [Fa0/4] FastEthernet0/4" /> <port id="577" descr=" [Fa0/5] FastEthernet0/5" /> <port id="578" descr=" [Fa0/6] FastEthernet0/6" /> </device> </C></pre>

The third example is the last call, where the number of available elements were less than **maxResults** parameters (the **elementsProcessed** is less than 5, and the returned **position** is "END"):

INPUT	curl -u admin:admin https://localhost/api/dataAccess/C?position=646:2,805:0&maxResults=5&view=All%20Objects&media=xml -H "content-type: application/xml"
OUTPUT	<pre><C serverName="ENTLONDEV08" serverId="dae26869-46e3-46df-91a0-28ce61b28d05" templateServerId="dae26869-46e3-46df-91a0-28ce61b28d05" startTime="0" endTime="0" elementsProcessed="4" position="END" processingTime="0"> <device id="646" name="e2821"> <port id="806" descr=" [Gi0/1] GigabitEthernet0/1-mpls layer" /> <port id="807" descr=" [Se0/1/0] Serial0/1/0-mpls layer" /> </device> </C></pre>

The GET call also accepts parameters that will be defined as templateParameters that can be used in expressions in the DataAccessTemplate. Any parameters passed in the URL that match an attribute name will be accessible in the statement language embedded in the template as 'templateParameter.<parameter value>'.

URL format: <parameter name>=<value>

Notes:

- the parameter's name must be an existing attribute name.
- arguments of type string, integer and float are supported.
- string values with embedded " or \ characters are escaped.
- string values may be optionally wrapped with quotes.

e.g.

name=myDevice

name="myDevice"

name='myDevice'

Example 1:

```
curl -u admin:admin "https://<entuity server>/api/dataAccess/A?view=All%20objects&id=804&media=xml"
```

A filter expression in the template:

```
<filter>simple;templateParameter.id==id</filter>
```

would match any objects with an attribute 'id' with the value '804'.

Example 2:

```
curl -u admin:admin "https://<entuity server>/api/dataAccess/B?view=All%20objects&name=myDevice&media=xml"
```

A filter expression in the template:

```
<filter>simple;templateParameter.name==name</filter>
```

would match any objects with an attribute 'name' with the value 'myDevice'.

Flow Data - Listing Devices with Stored Flow Information

Lists network devices for which flow is being collected and stored, and flow history for a specified device. Network devices must be configured to send flow information to ENA, meaning that ENA has no control over the flow information it receives. Further, flow information is not stored by default - this must be enabled for each device where desired.

URL: `http(s)://{server hostname}/api/flowDevices`

- [Method summary](#)
- [GET Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method summary

- [GET Method details](#) - lists devices for which flow information is being collected.

GET Method detail

Lists the devices for which flow information is currently being collected.

Request

Call the resource URL with the GET method.

Response

The API will respond with a list of devices for which ENA has collected flow data. Within the list, each device is represented as an associative array, containing the following information about the devices:

Name	Description
flowServerId	ID of the flow server that is collecting flow for this device. Typically, each Entuity server manages a single flow server. In this case you would expect flowServerId to match serverId . However, an Entuity server can manage multiple flow servers and this attribute would allow you to determine which one.
zoneName	name of the zone that the device has been placed in. If zoning is not in use, the value of zoneName will be null.
deviceId	ID of the device as assigned when the device is taken under management by ENA. This ID is unique to a server but must be used in conjunction with flowServerId to uniquely identify a device in a multi-server configuration.

IpAddress	string representation of the management IP address of the device. IPv4 and IPv6 are supported.
deviceName	string representation of the device's display name.
swObld	StormWorks Object ID. All objects, including devices, are assigned a stormworks object ID. As with deviceId , swObld must be used in conjunction with flowServerId to uniquely identify an object in a multi-server configuration.
vxlan	indicates that the flow record originates from a VMWare virtualized device. Either True or False.
ifIndexes	list of port ifIndexes for which flow is being collected. Note that ifIndexes are only unique within a device.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/flowDevices?media=json</code>
OUTPUT	<pre>{ "devices" : [{ "flowServerId" : "5ba829fa-86d9-4246-b4ce-24cc896180c0", "zoneName" : null, "deviceId" : 17, "ipAddress" : "10.18.0.2", "deviceName" : "ca01", "swObjId" : 866, "vxlan" : false, "ifIndexes" : [1, 2, 3, 4, 5] }, { "flowServerId" : "5ba829fa-86d9-4246-b4ce-24cc896180c0", "zoneName" : null, "deviceId" : 1, "ipAddress" : "10.1.2.2", "deviceName" : "hq01", "swObjId" : 730, "vxlan" : false, "ifIndexes" : [1, 2, 3, 4, 5] }] }</pre>

Flow Data - Listing Applications Supporting Flow

Lists the applications currently configured to support flow. ENA can characterize flow information by application, doing so by looking at the destination port at the transport layer (TCP/UDP).

URL: `http(s)://{server hostname}/api/api/flowApplications`

- [Method summary](#)
- [GET Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method summary

- GET Method details - lists applications configured to support flow.

GET Method detail

List applications configured to support flow.

Request

Call the resource URL with the GET method.

Response

An associative array containing the single member called "applications". The value of applications is a list of application names.

Examples

This list contains 5665 applications by default, so the list has been truncated in this example.

INPUT	<code>curl -u admin:admin https://testld.entuity.com/api/options/flowApplications?media=json</code>
-------	---

OUTPUT	<pre>{ "applications" : [{ "name" : "1ci-smcs" }, { "name" : "3Com-nsd" }, { "name" : "3PC" }, { "name" : "3com-amp3" }, { ... }]</pre>
---------------	---

Flow Data - Time Series History

Lists time series by flow data, grouped and filtered according to query parameters.

URL: `http(s)://{server hostname}/api/flowHistory`

- [Method summary](#)
- [GET Method detail](#)
 - [Request](#)
 - [Examples](#)

Method summary

- GET Method details - lists time series by flow data.

GET Method detail

List time series by flow data

Request

Call the resource URL with the GET method.

Examples

Total traffic for deviceID 1 at 5 minute intervals in last 30 minutes:

INPUT	<code>curl -u admin:admin https://localhost/api/flowHistory/1?media=xml&startTime=-1800&interval=300</code>
-------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="https://www.entuity.com/schemas/flow" > <sampleSet> <sample rate="9323.48" timestamp="1507541700" volume="2797044"/> <sample rate="12137.806666666667" timestamp="1507542000" volume="3641342"/> <sample rate="9914.983333333334" timestamp="1507542300" volume="2974495"/> <sample rate="8400.313333333334" timestamp="1507542600" volume="2520094"/> <sample rate="8922.28" timestamp="1507542900" volume="2676684"/> </sampleSet> </ns2:flowHistoryResult></pre>
---------------	---

Grouped by ingress interface:

INPUT	<pre>curl -u admin:admin https://localhost/api/flowHistory/1? media=xml&startTime=-900&interval=300&groups=ifIn</pre>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="https://www.entuity.com/schemas/flow"> <sampleSet> <key>5</key> <sample rate="5446.903333333334" timestamp="1507543200" volume="1634071"/> <sample rate="4787.08" timestamp="1507543500" volume="1436124"/> </sampleSet> <sampleSet> <key>7</key> <sample rate="3815.54" timestamp="1507543200" volume="1144662"/> <sample rate="3208.3" timestamp="1507543500" volume="962490"/> </sampleSet> </ns2:flowHistoryResult></pre>

Grouped by egress interface:

INPUT	<pre>curl -u admin:admin https://localhost/api/flowHistory/1? media=xml&startTime=-900&interval=300&groups=ifOut</pre>
--------------	--

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="https://www.entuity.com/schemas/flow"> <sampleSet> <key>0</key> <sample rate="1959.3333333333333" timestamp="1507543500" volume="587800"/> <sample rate="1614.19" timestamp="1507543800" volume="484257"/> </sampleSet> <sampleSet> <key>5</key> <sample rate="3177.6666666666665" timestamp="1507543500" volume="953300"/> <sample rate="4375.68" timestamp="1507543800" volume="1312704"/> </sampleSet> <sampleSet> <key>7</key> <sample rate="2858.38" timestamp="1507543500" volume="857514"/> <sample rate="3881.9966666666664" timestamp="1507543800" volume="1164599"/> </sampleSet> </ns2:flowHistoryResult> </pre>
---------------	--

Grouped by interface (note the double counting):

INPUT	<pre> curl -u admin:admin https://localhost/api/flowHistory/1? media=xml&startTime=-900&interval=300&groups=if </pre>
--------------	---

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="https://www.entuity.com/schemas/flow"> <sampleSet> <key>0</key> <sample rate="1614.19" timestamp="1507543800" volume="484257"/> <sample rate="2288.0533333333333" timestamp="1507544100" volume="686416"/> </sampleSet> <sampleSet> <key>5</key> <sample rate="9842.5633333333334" timestamp="1507543800" volume="2952769"/> <sample rate="10004.85" timestamp="1507544100" volume="3001455"/> </sampleSet> <sampleSet> <key>7</key> <sample rate="8286.98" timestamp="1507543800" volume="2486094"/> <sample rate="7776.7366666666667" timestamp="1507544100" volume="2333021"/> </sampleSet> </ns2:flowHistoryResult> </pre>
---------------	---

Grouped by application and sender. Restricted to limited set of applications and IP subnet :

INPUT	<pre> curl -u admin:admin https://localhost/api/flowHistory/1? media=xml&startTime=-900&interval=300&groups=appName,srcIP&filter=AND(OR (OR(EQ(appName,ICMP),EQ(appName,snmp)),EQ(appName,smtp)),LIKE(srcIP, 10.44.2.*)) </pre>
--------------	---

OUTPUT	<pre> <?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="https://www.entuity.com/schemas/flow"> <sampleSet> <key>ICMP </key> <key>10.44.2.10</key> <sample rate="0.32" timestamp="1507559400" volume="96"/> <sample rate="0.48" timestamp="1507559700" volume="144"/> </sampleSet> <sampleSet> <key>ICMP</key> <key>10.44.2.76</key> <sample rate="0.32" timestamp="1507559400" volume="96"/> <sample rate="0.48" timestamp="1507559700" volume="144"/> </sampleSet> <sampleSet> <key>smtp</key> <key>10.44.2.130</key> <sample rate="1.0133333333333334" timestamp="1507559400" volume="304"/> <sample rate="1.52" timestamp="1507559700" volume="456"/> </sampleSet> <sampleSet> <key>snmp</key> <key>10.44.2.122</key> <sample rate="140.92666666666668" timestamp="1507559400" volume="42278" /> <sample rate="160.25" timestamp="1507559700" volume="48075"/> </sampleSet> </ns2:flowHistoryResult> </pre>
---------------	---

Maintenance List

Lists available maintenance schedules or create a new maintenance schedule.

URL: maintenance

- [Method Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Example](#)
- [POST Method detail](#)
 - [Request](#)
 - [Example](#)

Method Summary

- GET Method - lists available maintenance schedules.
- POST Method - create a new maintenance schedule.

GET Method detail

Lists available maintenance schedules.

Response

A map of maintenance ids to their names.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/maintenance</code>
-------	--

OUTPUT	<pre> { "items" : [{ "id" : 29, "name" : "supp" }, { "id" : 30, "name" : "test" }], "count" : 2 }</pre>
---------------	---

POST Method detail

Creates a new maintenance schedule.

Request

A maintenance settings object. This object is used many times in Maintenance API.

Name	Description
id	maintId of this maintenance schedule.
name	specified name of this maintenance schedule. Must not be empty or null, and must be unique for this server. Maximum of 255 characters.
description	specified description of this maintenance schedule. Maximum of 255 characters.
devices	Map of serverIds (string) to a List of deviceIds (integer). Contains the devices that will be affected by this schedule. Devices are identified by their deviceId on the server from the serverId. This value will be ignored if it is not used in a POST call to create a new schedule.
suppressEvents	if true , then the devices will have their events suppressed during the times that they would be under maintenance.
serverId	serverId of the server that created this schedule.
masterId	if 0, this schedule was created locally. Otherwise, the serverId is the master of this schedule, and the original schedule's ID is this value.
addDevices	map of serverIds to list of deviceIds to add to the existing list of devices. Only applicable to modifying existing schedules, otherwise will be null.
removeDevices	map of serverIds to list of deviceIds to remove from the existing list of devices. Only applicable to modifying existing schedules, otherwise will be null.
schedule	an object that describes a schedule.
begins	time in which the schedule starts in seconds.

ends	time in which the schedule stops in seconds.
recurs	if false , the schedule will be active throughout the time between "begins" and "ends". If true , follows a recurrence of periods given by the below values, but only between "begins" and "ends".
daysSecondsFrom	time of day the maintenance schedule should start being active in seconds. (Hours * 3600 + Minutes * 60 + Seconds) (e.g. 0 = 0000 hours (12 AM), 3600 = 0100 (1 AM), 34200 = 0930 (9:30AM)) If "daysSecondsFrom" = 0 and "daysSecondsTo" = 86400 (24 hours), then it will be active the whole day.
daysSecondsTo	time of day it should stop being active in seconds. Must be larger than "daysSecondsFrom".
recurrenceKind	can be "EVERY_DAY", "WEEK_DAYS", "MONTH_DAYS".
validDays	string description of valid days. Section of days are separated by a comma ",", and if there are 2 or more consecutive days then the first and last days are given with a hyphen "-" in between (e.g. "1, 5, 6-7, 10-22"). When setting this field, it is valid to contain consecutive days separately. If there are values greater than the maximum, those values will be ignored (e.g. 8 for "WEEK_DAYS"). If "recurrenceKind" is "WEEK_DAYS", this describes the week days on which the schedule should be active (max 7). (e.g. 1 = Monday, 7 = Sunday). If "recurrenceKind" is "MONTH_DAYS", this describes the days in a month on which the schedule should be active (max 31) (e.g. 3 = 3rd of every month, 31 = 31st of every month that has a 31st day).
validMonths	string description of valid months (e.g. 1 = January, 6 = June).
invert	if true , this maintenance will be active when outside of the given schedule.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/maintenance -X POST -H "content-type:application/json" -d \ { "name" : "ddd", "devices" : { "server-id" : [12] }, "suppressEvents" : false, "schedule" : { "begins" : 1562910269, "ends" : 0, "recurs" : true, "daysSecondsFrom" : 39600, "daysSecondsTo" : 50400, "recurrenceKind" : "MONTH_DAYS", "validDays" : "2-3, 8, 10, 15, 17-18, 22-31", "validMonths" : "1-4, 6-12", } }</pre>
-------	---

```
OUTPUT {  
  "id" : 5,  
  "name" : "ddd",  
  "description" : "",  
  "devices" : {  
    "test" : [ 12 ]  
  },  
  "suppressEvents" : false,  
  "schedule" : {  
    "begins" : 1562910269,  
    "ends" : 0,  
    "recurs" : true,  
    "daysSecondsFrom" : 39600,  
    "daysSecondsTo" : 50400,  
    "recurrenceKind" : "MONTH_DAYS",  
    "validDays" : "2-3, 8, 10, 15, 17-18, 22-31",  
    "validMonths" : "1-4, 6-12",  
    "invert" : false  
  },  
  "serverId" : "c4237d15-de15-4d03-9e07-2a76ddd95018",  
  "masterId" : 0,  
  "addDevices" : null,  
  "removeDevices" : null  
}
```

Maintenance Details

Inspect, update or delete a maintenance schedule.

URL: maintenance/{maintenance id}

- [Method Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [Delete Method detail](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - inspect a maintenance schedule.
- PUT Method - update a maintenance schedule.
- DELETE Method - delete a maintenance schedule.

GET Method detail

Return all details of this maintenance schedule with the given maintenance id.

Response

Returns a Maintenance Settings object. See Maintenance Settings.

Name	Description
id	maintId of this maintenance schedule.
name	specified name of this maintenance schedule. Must not be empty or null, and must be unique for this server. Maximum of 255 characters.
description	specified description of this maintenance schedule. Maximum of 255 characters.
devices	map of serverIds (string) to a List of deviceIds (integer). Contains the devices that will be affected by this schedule. Devices are identified by their deviceId on the server from the serverId. This value will be ignored if it is not used in a POST call to create a new schedule.

suppressEvents	if true , then the devices will have their events suppressed during the times that they would be under maintenance.
serverId	serverId of the server that created this schedule.
masterId	if 0, this schedule was created locally. Otherwise, the serverId is the master of this schedule, and the original ID is this value.
addDevices	map of serverIds to list of deviceIds to add to the existing list of devices. Only applicable to modifying existing schedules, otherwise will be null.
removeDevices	map of serverIds to list of deviceIds to remove from the existing list of devices. Only applicable to modifying existing schedules, otherwise will be null.
schedule	an object that describes a schedule.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/maintenance/2?media=json</code>
--------------	---

OUTPUT	<pre> { "id" : 2, "name" : "test", "description" : "ate", "devices" : { }, "suppressEvents" : true, "schedule" : { "begins" : 1571838648, "ends" : 1572965448, "recurs" : true, "daysSecondsFrom" : 0, "daysSecondsTo" : 86400, "recurrenceKind" : "WEEK_DAYS", "validDays" : "5-6", "validMonths" : "1-4, 6, 8-9, 11-12", "invert" : false }, "serverId" : "46e4aed7-3804-494a-90f8-3867481a3d39", "masterId" : 0, "addDevices" : null, "removeDevices" : null }</pre>
---------------	---

PUT Method detail

Modify this maintenance schedule with given values to the schedule with the given maintenance id, and return the modified maintenance schedule.

Request

Returns a Maintenance Settings object. See [Maintenance Settings](#).

Name	Description
id	maintId of this maintenance schedule.
name	specified name of this maintenance schedule. Must not be empty or null, and must be unique for this server. Maximum of 255 characters.

description	specified description of this maintenance schedule. Maximum of 255 characters.
devices	map of serverIds (string) to a List of deviceIds (integer). Contains the devices that will be affected by this schedule. Devices are identified by their deviceId on the server from the serverId. This value will be ignored if it is not used in a POST call to create a new schedule.
suppressEvents	if true , then the devices will have their events suppressed during the times that they would be under maintenance.
serverId	serverId of the server that created this schedule.
masterId	if 0, this schedule was created locally. Otherwise, the serverId is the master of this schedule, and the original ID is this value.
addDevices	map of serverIds to list of deviceIds to add to the existing list of devices.
removeDevices	map of serverIds to list of deviceIds to remove from the existing list of devices.
schedule	an object that describes a schedule.

Response

Returns a Maintenance Settings object. See [Maintenance Settings](#).

The devices list cannot be set with this method, and any changes must be made using "addDevices" and "removeDevices".

Any value that does not exist in the request will keep its original value. However, **schedule** is an object, so any changes to schedule will require all values in the schedule to be present.

Name	Description
id	maintId of this maintenance schedule.
name	specified name of this maintenance schedule. Must not be empty or null, and must be unique for this server. Maximum of 255 characters.
description	specified description of this maintenance schedule. Maximum of 255 characters.
devices	map of serverIds (string) to a List of deviceIds (integer). Contains the devices that will be affected by this schedule. Devices are identified by their deviceId on the server from the serverId. This value will be ignored if it is not used in a POST call to create a new schedule.
suppressEvents	if true , then the devices will have their events suppressed during the times that they would be under maintenance.
serverId	serverId of the server that created this schedule.
masterId	if 0, this schedule was created locally. Otherwise, the serverId is the master of this schedule, and the original ID is this value.
addDevices	map of serverIds to list of deviceIds to add to the existing list of devices.
removeDevices	map of serverIds to list of deviceIds to remove from the existing list of devices.
schedule	an object that describes a schedule.

Examples

INPUT	<pre>curl -u admin:admin https://localhost/api/maintenance/2 -X PUT -H "content-type:application/json" -d \ ' { "addDevices" : { "server-id" : [12] } }'</pre>
OUTPUT	<pre>{ "id" : 2, "name" : "test", "description" : "ate", "devices" : { "server-id" : [12] }, "suppressEvents" : true, "schedule" : { "begins" : 1571838648, "ends" : 1572965448, "recurs" : true, "daysSecondsFrom" : 0, "daysSecondsTo" : 86400, "recurrenceKind" : "WEEK_DAYS", "validDays" : "5-6", "validMonths" : "1-4, 6, 8-9, 11-12", "invert" : false }, "serverId" : "46e4aed7-3804-494a-90f8-3867481a3d39", "masterId" : 0, "addDevices" : null, "removeDevices" : null }</pre>

Delete Method detail

Remove this maintenance schedule. Returns OK if successful.

Response

Returns OK status if successful.

Example

INPUT	<code>curl -X DELETE -u admin:admin https://localhost/api/maintenance/2?media=json</code>
OUTPUT	<pre>{ "message" : "OK" }</pre>

Meraki Cloud Controllers

Lists Meraki Cloud Controllers

URL: `http(s)://{server hostname}/api/MerakiCloudControllers?serverId={serverId}`

If the serverId is left blank, then the server will be the one being contacted.

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists Meraki Cloud Controllers.

GET Method details

Lists Meraki Cloud Controllers

Response

The list of items, with entries according to the following format:

Name	Description
serverID	the Entuity server ID string.
objectID	the StormWorks object ID of the Meraki device on the server.
name	the display name of the device in ENA.
stormWorksType	StormWorks type.
webhookURL	this is the URL that should be put in the settings of the cloud controller (using the Cisco website, not anything in ENA).
secretKeysSet	if the secret key is set.
sharedSecret	if the secret key is shared.
webhookEnabled	if the webhook is enabled.
status	this is the same as the status displayed in the ENA webhook admin page.

merakiDeviceCount	number of devices managed by the cloud controller.
secretKey	key that should be entered on the cloud controller settings page (same as the URL).

Examples

INPUT	curl -u admin:admin https://localhost/api/MerakiCloudControllers
OUTPUT	<pre>{ "items" : [{ "serverId" : "9a55e715-3c18-4ef1-9cc9-f1b7f29ea139", "objectId" : 5073, "name" : "Meraki", "stormWorksType" : "MerakiCloudController", "webhookURL" : "https://localhost/webUI/api/webhook/cloudAlert/Meraki?serverId=9a55e715-3c18-4ef1-9cc9-f1b7f29ea139&objectId=5073", "secretKeyIsSet" : false, "sharedSecret" : null, "webhookEnabled" : true, "status" : "0 alerts in the last 24hrs", "merakiDeviceCount" : 109 }], "count" : 1 }</pre>

Shared Secret Key

List, create or delete shared secret keys

URL: `http(s)://{server hostname}/api/webhook/cloudAlert/Meraki/secretKey?objectId={StormWorks object ID}&serverId={server ID}`

objectId is the StormWorks ID of the Meraki Cloud Controller. You must provide a StormWorks object ID.

serverId is an optional parameter.

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - list shared secret keys
- POST Method - create shared secret keys
- DELETE Method - delete shared secret keys

GET Method details

Lists shared secret keys

Response

The list of items, with entries according to the following format:

Name	Description
serverID	the Entuity server ID string.
objectId	the StormWorks object ID of the Meraki device on the server.
name	the display name of the device in ENA.
stormWorksType	StormWorks type.
webhookURL	this is the URL that should be put in the settings of the cloud controller (using the Cisco website, not anything in ENA).

secretKeysSet	if the secret key is set.
sharedSecret	if the secret key is shared.
webhookEnabled	if the webhook is enabled.
status	this is the same as the status displayed in the ENA webhook admin page.
merakiDeviceCount	number of devices managed by the cloud controller.
secretKey	key that should be entered on the cloud controller settings page (same as the URL).

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhook/cloudAlert/Meraki/secretKey?objectId=5073</code>
OUTPUT	<pre>{ "serverId" : "9a55e715-3c18-4ef1-9cc9-f1b7f29ea139", "objectId" : 5073, "secretKey" : null }</pre>

POST Method details

Adds shared secret keys

Response

The list of items, with entries according to the following format:

Name	Description
serverID	the Entuity server ID string.
objectId	the StormWorks object ID of the Meraki device on the server.
name	the display name of the device in ENA.
stormWorksType	StormWorks type.
webhookURL	this is the URL that should be put in the settings of the cloud controller (using the Cisco website, not anything in ENA).
secretKeysSet	if the secret key is set.
sharedSecret	if the secret key is shared.
webhookEnabled	if the webhook is enabled.
status	this is the same as the status displayed in the ENA webhook admin page.

merakiDeviceCount	number of devices managed by the cloud controller.
secretKey	key that should be entered on the cloud controller settings page (same as the URL).

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhook/cloudAlert/Meraki/secretKey?objectId=5073 -X POST</code>
OUTPUT	<pre>{ "serverId" : "9a55e715-3c18-4ef1-9cc9-f1b7f29ea139", "objectId" : 5073, "secretKey" : "bc6241ab-6481-4c5a-8d42-6369a199895c" }</pre>

DELETE Method details

Deletes shared secret keys

Response

The list of items, with entries according to the following format:

Name	Description
serverID	the Entuity server ID string.
objectId	the StormWorks object ID of the Meraki device on the server.
name	the display name of the device in ENA.
stormWorksType	StormWorks type.
webhookURL	this is the URL that should be put in the settings of the cloud controller (using the Cisco website, not anything in ENA).
secretKeysSet	if the secret key is set.
sharedSecret	if the secret key is shared.
webhookEnabled	if the webhook is enabled.
status	this is the same as the status displayed in the ENA webhook admin page.
merakiDeviceCount	number of devices managed by the cloud controller.
secretKey	key that should be entered on the cloud controller settings page (same as the URL).

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhook/cloudAlert/Meraki/secretKey?objectId=5073 -X DELETE</code>
OUTPUT	<pre>{ "serverId" : "9a55e715-3c18-4ef1-9cc9-f1b7f29ea139", "objectId" : 5073, "secretKey" : "" }</pre>

Meraki Webhooks

Return the status of, and enable or disable, Meraki webhooks.

URL: `http(s)://{server hostname}/api/webhook/cloudAlert/Meraki/enabled?objectId={StormWorks object ID}&serverId={server ID}`

objectId is the StormWorks ID of the Meraki Cloud Controller. You must provide a StormWorks object ID.

serverId is an optional parameter.

- [Methods Summary](#)
- [GET Method details](#)
 - [Example](#)
- [PUT Method details](#)
 - [Example](#)

Methods Summary

- GET Method - return the status of a Meraki webhook, i.e. enabled or disabled.
- PUT Method - enable or disable a Meraki webhook. Will accept true (enable) or false (disable).

GET Method details

Return the status of a Meraki webhook, i.e. enabled or disabled.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhook/cloudAlert/enabled/objectId=5073</code>
OUTPUT	true

PUT Method details

Enable or disable a Meraki webhook. Will accept true (enable) or false (disable).

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhook/cloudAlert/enabled/objectId=5073 -X PUT -H "Content-type:plain/text" -d "false"</code>
OUTPUT	false

Cisco DNA Center Webhooks

List DNA Center webhooks.

URL: `http(s)://{server hostname}/api/webhook/cloudAlert`

- [Methods Summary](#)
- [GET Method details](#)
 - [Example](#)

Methods Summary

- GET Method - list DNA Center webhooks

GET Method details

Returns a list of DNA Center webhooks.

Example

INPUT	<code>curl -k -u admin:admin https://localhost/api/webhook</code>
OUTPUT	<pre>{ "items" : [{ "serverId" : "163b10d7-be44-4c7b-9b6c-db692a1c87d1", "objectId" : 749, "name" : "DNAC1", "stormWorksType" : "DNACenter", "webhookURL" : "https://neon/api/webhook/cloudAlert/DNAC/749?serverId=163b10d7-be44-4c7b-9b6c-db692a1c87d1", "webhookEnabled" : true, "status" : "28 alerts in the last 24hrs", "secretKeyIsSet" : false, "sharedSecret" : null, "deviceCount" : null }], "count": 1 }</pre>

Raise Webhook Events on Cisco DNA Centers

Enable or disable webhook events on a DNA Center.

URL: `http(s)://{server hostname}/api/webhook/cloudAlert/DNAC/{ObjectID}?serverId={serverId}`

- [Methods Summary](#)
- [PUT Method details](#)
 - [Example](#)

Methods Summary

- PUT Method - enable or disable webhook events on a DNA Center.

PUT Method details

Enable or disable webhook events on a DNA Center.

Example

INPUT	<pre> curl -k -X POST -u admin:admin https://localhost/api/webhook/cloudAlert /DNAC/<ObjectID>?serverId=<serverId> -H "content-type:application /json" -d \ ' { "version": null, "instanceId": "8169d18b-ef31-4163-8d90-d9e535b55fb5", "eventId": "NETWORK-NON-FABRIC_WIRED-1-200", "namespace": "ASSURANCE", "name": null, "description": null, "type": "NETWORK", "category": "ALERT", "domain": "Connectivity", "subDomain": "Non-Fabric Wired", "severity": 1, "source": "ndp", "timestamp": 1583315649978, "tags": ["tag1", "tag2"], "details": { "Type": "Network Device", "Assurance Issue Details": "This network device ABC is unreachable from controller. The device role is EDGE", "Assurance Issue Priority": "P1", "Device": "1.2.3.4", "Assurance Issue Name": "Network Device 2.3.4.5 Is Unreachable From Controller", "Assurance Issue Category": "Availability", "Assurance Issue Status": "active" }, "ciscoDnaEventLink": "dna/assurance/issueDetails?issueId=8169d18b- ef31-4163-8d90-d9e535b55fb5", "note": "To programmatically get more info see here - https://<ip- address>/dna/platform/app/consumer-portal/developer-toolkit/apis? apiId=8684-39bb-4e89-a6e4", "tntId": "", "context": null, "tenantId": "" }' </pre>
-------	---

OUTPUT (when Webhook is enabled)	<pre>{ "message" : "Success. DNA Center Alert Webhook received and Entuity event raised." }</pre>
OUTPUT (when Webhook is disabled)	<pre>{ "code" : 403, "contactEmail" : null, "description" : "Webhook is not enabled for this DNA Center", "homeRef" : "/", "reasonPhrase" : "Forbidden", "uri" : "https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.4" }</pre>

Manage Cisco DNA Center Webhooks

Return the status of, and enable or disable, Cisco DNA Center webhooks.

URL: `http(s)://{server hostname}/api/webhook/cloudAlert`

- [Methods Summary](#)
- [GET Method details](#)
 - [Example](#)
- [PUT Method details](#)
 - [Example](#)

Methods Summary

- GET Method - return the status of a Cisco DNA Center webhook flag for objectID, i.e. enabled or disabled.
- PUT Method - enable or disable a Cisco DNA Center webhook flag for objectID. Will accept true (enable) or false (disable).

GET Method details

Return the status of a Cisco DNA Center webhook flag for objectID, i.e. enabled or disabled.

Example

INPUT	<code>curl -u admin:admin -k https://localhost/api/webhook/cloudAlert/enabled/<objectId></code>
OUTPUT	false

PUT Method details

Enable or disable a Cisco DNA Center webhook flag for objectID. Will accept true (enable) or false (disable).

Example

INPUT	<code>curl -u admin:admin -k https://localhost/api/webhook/cloudAlert/enabled/<objectId> -X PUT -d "false"</code>
OUTPUT	false

Cisco DNA Centers

Return a list of DNA Centers.

URL: `http(s)://{server hostname}/api/DNACenters`

- [Methods Summary](#)
- [GET Method details](#)
 - [Example](#)

Methods Summary

- GET Method - return a list of DNA Centers.

GET Method details

Return a list of DNA Centers.

Example

INPUT	<code>curl -k -u admin:admin https://localhost/api/DNACenters</code>
OUTPUT	<pre>{ "items" : [{ "serverId" : "163b10d7-be44-4c7b-9b6c-db692a1c87d1", "objectId" : 749, "name" : "DNAC1", "stormWorksType" : "DNACenter", "webhookURL" : "https://enaserver/api/webhook/cloudAlert/DNAC/749?serverId=163b10d7-be44-4c7b-9b6c-db692a1c87d1", "webhookEnabled" : false, "status" : "28 alerts in the last 24hrs", "secretKeyIsSet" : false, "sharedSecret" : null, "deviceCount" : null }], "count": 1 }</pre>

Credential Management

List, create, edit or remove credentials on a server. Applicable to Enuity v19.0 upwards only.

URL: credential

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Response](#)
 - [Examples](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists credentials.
- POST Method - creates a new credential.
- PUT Method - edit a credential.
- DELETE Method - remove a credential.

GET Method detail

Lists credentials on a server.

Response

Response includes a list of credentials. Each credential has the following attributes.

Examples

INPUT	<code>curl -u admin:admin -k https://localhost/api/credential?media=json</code>
-------	---

OUTPUT	<pre>{ "items" : [{ "id" : 1, "deviceId" : 0, "kind" : "SHARED", "name" : "Public modified", "description" : "modified", "attributes" : { "SnmpV12Attributes" : { "community" : "*****", "writeCommunity" : "*****", } } }], "count" : 1 }</pre>
---------------	--

INPUT	curl -u admin:admin https://localhost/api/credential?media=xml
OUTPUT	<pre>{ "items" : [{ "id" : 18, "deviceId" : 0, "kind" : "SHARED", "name" : "Credential 1", "description" : "This is a credential", "attributes" : { "SnmpV2Attributes" : { "community" : "*****", "writeCommunity" : "*****" } } }, { "id" : 19, "deviceId" : 0,</pre>

```

    "kind" : "SHARED",
    "name" : "Credential 2",
    "description" : "This is also a credential",
    "attributes" : {
      "SshAttributes" : {
        "username" : "admin",
        "password" : "*****",
        "sshKey" : "*****"
      }
    }
  }, {
    "id" : 20,
    "deviceId" : 0,
    "kind" : "SHARED",
    "name" : "Third Cred",
    "description" : "Cred number 3",
    "attributes" : {
      "NaviSecAttributes" : {
        "scope" : "GLOBAL",
        "username" : "admin",
        "password" : "*****"
      }
    }
  } ],
  "count" : 3
}

```

POST Method details

Creates a new credential. The user must know the type of credential required. Entuity supports several types of credential and each has different attributes, which is also applicable to the PUT Method.

Request

SnmpV1Attributes; SnmpV2Attributes; SnmpV12Attributes:

Name	Description
community	Community string.
writeCommunity	Write community string.

SnmpV3Attributes:

Name	Description
username	V3 username.
authType	Supported types: • NONE • MD5 • SHA • SHA224 • SHA256 • SHA384 • SHA512.
authPassword	Authentication password.
encryptionType	Supported types: • NONE • DES • TRIPLE_DES • AES • AES256.
encryptionPassword	Encryption password.
context	optional V3 context.

CLiAccessAttributes (Config Management credential):

Name	Description
method	Supported types are: • SSH • TELNET.
port	The port to connect.
username	Username if required.
password1	Optional password one.

password2	Optional password two.
------------------	------------------------

UserAndPasswordAttributes:

Name	Description
username	Username.
password	password.

SshAttributes:

Name	Description
username	Username.
password	password.
sshKey	SSH key. Only one of the password or SSH key must be specified.

NaviSecAttributes:

Name	Description
scope	Supported scopes are: • GLOBAL • LOCAL • LDAP.
username	connection username.
password	connection password.

AzureAttributes:

Name	Description
clientId	Unique client identity.
secretKey	Azure vault secret key.
tenantId	Tenant identity (GUID).

AwsAttributes:

Name	Description
accessKey	Access Key.
secretKey	Secret Key.

ApiKeyAttributes:

Name	Description
key	Meraki access key.

Response

The list of credentials after an update, as GET method would return them.

Examples

1. Creating an SNMPv1 credential.

INPUT	<pre>curl -u admin:admin -k https://localhost/api/credential?media=json -X "POST" -H "content-type:application/json" -d '{ "name":"0: RestAPI Test-With Description", "description":"Test description", "attributes":{ "SnmpV1Attributes":{ "community":"public", "writeCommunity":"write" } } }'</pre>
--------------	---

OUTPUT	<pre>{ "id" : 3042, "deviceId" : 0, "kind" : "SHARED", "name" : "0: RestAPI Test-With Description", "description" : "Test description", "attributes" : { "SnmpV1Attributes" : { "community" : "*****", "writeCommunity" : "*****" } } }</pre>
---------------	---

2. Creating a SSH credential.

INPUT	<pre>curl -u admin:admin -k https://localhost/api /credential?media=json -X "POST" -H "content-type: application/json" -d '{" "name":"178: RestAPI Test SSH-With Description", "description":"Test description with SSH", "attributes":{ "CliAccessAttributes":{ "method":"SSH", "port":"22", "username":"test", "password1":"test!test", "password2":"test!test" } } }'</pre>
--------------	--

OUTPUT	<pre>{ "id" : 449, "deviceId" : 0, "kind" : "SHARED", "name" : "178: RestAPI Test SSH-With Description", "description" : "Test description with SSH", "attributes" : { "CliAccessAttributes" : { "method" : "SSH", "port" : 22, "username" : "test", "password1" : "*****", "password2" : "*****" } } }</pre>
---------------	---

3. Creating a V3 credential.

INPUT	<pre>curl -u admin:admin -k https://localhost/api/credential?media=json -X "POST" -H "content-type:application/json" -d '{ "name":"4: RestAPI Test NONE-NONE No Auth/Encr Pass-With Description", "description":"Test description NONE-NONE", "attributes":{ "SnmpV3Attributes":{ "username":"test", "authType":"NONE", "authPassword":"", "encryptionType":"NONE", "encryptionPassword":"" } } }'</pre>
--------------	--

OUTPUT	<pre>{ "id" : 108, "deviceId" : 0, "kind" : "SHARED", "name" : "4: RestAPI Test NONE-NONE No Auth/Encr Pass-With Description", "description" : "Test description NONE-NONE", "attributes" : { "SnmpV3Attributes" : { "username" : "test", "authType" : "NONE", "authPassword" : "*****", "encryptionType" : "NONE", "encryptionPassword" : "*****" } } }</pre>
---------------	--

4. Creating a CLI credential.

INPUT	<pre>curl -u admin:admin -k https://localhost/api/credential?media=json -X "POST" -H "content-type:application/json" -d '{ "name":"181: RestAPI Test SSH No Port-Dupe", "attributes":{ "CliAccessAttributes":{ "method":"SSH", "port":"", "username":"test no port", "password1":"test", "password2":"test" } } }'</pre>
--------------	--

OUTPUT	<pre>{ "id" : 464, "deviceId" : 0, "kind" : "SHARED", "name" : "181: RestAPI Test SSH No Port-Dupe", "attributes" : { "CliAccessAttributes" : { "method" : "SSH", "port" : 0, "username" : "test no port", "password1" : "*****", "password2" : "*****" } } }</pre>
---------------	---

PUT Method details

Modifies parameters of the credential. Entuity supports several types of credential and each has different attributes - [please see the POST Method section above for the full list](#).

Response

Detailed information about the credential after changes, as GET method would return it.

Examples

Where <ID> is the credential ID. Users need only specify the attributes that are to be modified.

INPUT	<pre>curl -u admin:admin -k https://localhost/api /credential?media=json -X "PUT" -H "content-type: application/json" -d '{ "name" : "0: RestAPI Test-With Description modified" }' "https://localhost/api/credential/3042?media=json"</pre>
--------------	--

OUTPUT	<pre>{ "id" : 3042, "deviceId" : 0, "kind" : "SHARED", "name" : "0: RestAPI Test-With Description modified" }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin -k https://localhost/api/credential?media=json -X "PUT" -H "content-type:application/json" -d '{' "attributes" : { "SnmpV1Attributes" : { "community" : "new_value_test" } } }'</pre>
OUTPUT	<pre>{ "id" : 3515, "deviceId" : 0, "kind" : "SHARED", "attributes" : { "SnmpV1Attributes" : { "community" : "*****", "writeCommunity" : "*****" } } }</pre>

DELETE Method details

Deletes a credential.

Response

The current list of credentials, as the GET method would return it.

Examples

Where <ID> is the credential ID.

INPUT	<code>curl -u admin:admin -k https://localhost/api/credential/147?media=json -X "DELETE"</code>
OUTPUT	<pre>{ "message" : "Deleted credential Id 147" }</pre>

Services Hierarchy

List the services on a View, or create a service on a View.

URL: service

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Example](#)
- [POST Method detail](#)
 - [Request Parameters](#)
 - [Example](#)

Method Summary

- GET Method - list the services on a View.
- POST Method - create a service on a View.

GET Method detail

List the services on a View.

Request Parameters

All parameters are optional.

As with query strings, separate parameters with '&', e.g. 'indirect=true&subservices=true'.

Because it is a URL, spaces must be replaced with '%20', e.g. 'My%20Network' instead of 'My Network'.

Name	Description
viewpath	the View path to get services from. Should start with 'My%20Network' (HTML, URL encoding). If this value is not set, then the View path will default to 'All Objects', which would return all services. To access other users' View paths, prefix with the username and '::', e.g. 'user::My%20Network/AllObjects' Example: viewpath=My%20Network/test

indirect	if true, will also return services in child Views of the given View. This does not apply to consolidated calls. Example: indirect=true
subservices	if true, will also include subservices associated with retrieved services. This does not apply to consolidated calls. Example: subservices=true
consolidate	if true, will also retrieve service information from remote servers. Example: consolidate=true
maxdepth	maximum depth of child services to retrieve. If not set, this will default to 3. This applies only to consolidate=true. Example: maxdepth=5

Response data keys

Name	Description
objectId	object ID of this service. This is also referenced as 'serviceid' in other RESTful API calls.
serverObjectId	server ID of the server on which this service exists.
serviceName	user-specified name of service.
shortServiceName	user-specified short name of the service.
descriptiveAlias	user-specified description of service.
serviceStatus	gives the current status of the service. • -1 None • 0 Down • 1 Up • 2 Unknown • 3 Degraded
children	lists any children services associated with this service.
subServices	lists any subservices associated with this service.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/service</code>
--------------	--

OUTPUT	<pre>{ "value": [{ "objectId": 819, "serverId": "1758b1ea-2e6e-4bb2-82db-810a153293a2", "serviceName": "test", "shortServiceName": "", "descriptiveAlias": "", "serviceStatus": 1, "children": null "subServices": null } { "objectId": 825, "serverId": "1758b1ea-2e6e-4bb2-82db-810a153293a2", "serviceName": "ok", "shortServiceName": "", "descriptiveAlias": "", "serviceStatus": 1, "children": null "subServices": null }], "count": 2 }</pre>
---------------	---

INPUT	<pre>curl -u admin:admin https://localhost/api/service?viewpath=My%20Network/te /te2&indirect=true&subservices=true</pre>
--------------	---

```

OUTPUT {
  "value": [
    {
      "objectId": 819,
      "serverId": "1758b1ea-2e6e-4bb2-82db-810a153293a2",
      "serviceName": "test",
      "shortServiceName": "",
      "descriptiveAlias": "",
      "serviceStatus": 1,
      "children": null
      "subServices": null
    }
  ],
  "count": 1
}

```

POST Method detail

Create a new service on the selected View. Note that new services created this way do not contain any components, and must have them added via Service

Request Parameters

Name	Description
serviceName	user-specified name of service.
serviceType	type of service logic: • 0 None • 1 And • 2 Or • 3 Not • 4 At Least
serviceAtLeastValue	if the service logic type (serviceType) is 4 At Least, the serviceAtLeastValue specifies the minimum number of components that should be OK for the service to be considered OK.
serviceDegradedThreshold	value in which the service will be considered to be degraded.
raiseEvents	if the service will raise events upon a state change, either true or false .

treatUnknownAsDown	if the service will treat Unknown statuses as Down, either true or false .
serviceSlaGoal	SLA Availability goal in % (e.g. 99.0 = 99%).
descriptiveAlias	user-specified description of service.
parentViewPath	not applicable when creating a service (will use the #viewId given). The View path of the service.
parentServiceId	the parent service's ID.
aggregateEnabled	if traffic across all ports is aggregated.
serviceTag	service tag for reports. Can be: • 'Standard' • 'Branch' • 'CIO'
shortServiceName	user-specified short name of service.

All values except for serviceName can be left out and required fields will use the Default when Create value instead.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/service?viewpath=My%20Network/te -X POST -H "content-type:application/json" -d { "serviceName": "test" }</pre>
--------------	---

```
OUTPUT {
  "info": {
    "serviceId": 819,
    "serviceName": "test",
    "serviceType": 0,
    "serviceAtLeastValue": 0,
    "serviceDegradedThreshold": 0,
    "raiseEvents": true,
    "serviceSlaGoal": 99.0,
    "serviceWebImage": 0,
    "descriptiveAlias": "",
    "ownerId": "admin",
    "hasAdminPermission": true,
    "availableEyeServers": null,
    "users": [
      {
        "first": 3,
        "second": "admin",
      },
      {
        "first": 4,
        "second": "user",
      }
    ],
    "aggregateEnable": false,
    "serviceTag": "Standard"
  },
  "message": "ok"
},
"componentIds": [
  ]
}
```

Service Detail

List, modify, add components to or delete a service.

URL: service/{serviceId}

- [Method Summary](#)
- [GET Method detail](#)
 - [Response data keys](#)
 - [Example](#)
- [PUT Method detail](#)
 - [Request Parameters](#)
 - [Example](#)
- [POST Method detail](#)
 - [Request Parameters](#)
 - [Response Parameters](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - list details of a service.
- PUT Method - modifies settings of the service.
- POST Method - add or remove components to the service.
- DELETE Method - deletes the service.

GET Method detail

List details of a service.

Response data keys

Name	Description
serviceId	service ID of the service.
serviceName	user-specified service name.

serviceType	type of service logic: • 0 None • 1 And • 2 Or • 3 Not • 4 At Least
serviceAtLeastValue	if the service logic type (serviceType) is 4 At Least, the serviceAtLeastValue specifies the minimum number of components that should be OK for the service to be considered OK.
serviceDegradedThreshold	if the service logic type (serviceType) is 4 At Least, the serviceDegradedThreshold value specifies the value at which the service will be considered to be degraded.
raiseEvents	whether the service will raise events upon state change, either true or false .
treatUnknownAsDown	whether the service will treat Unknown statuses as Down, either true or false .
serviceSLAGoal	SLA availability goal in % (e.g. 99.0 = 99%)
serviceWebImage	not used.
descriptiveAlias	user-specified description of service.
ownerId	string ID of the user who owns the service.
hasAdminPermission	if the service was created by an Admin user (meaning that the service has access to components that may be admin only), either true or false .
availableEyeServerId	list of external (remote) servers that the server accesses.
defaultEyeServerId	default server from list of external (remote) servers.
users	list of users that have access to this service. Consists of the username and their ID.
aggregateEnabled	if traffic across all ports is aggregated.
serviceTag	service tags for reports. Can be: • "Standard" • "Branch" • "CIO"
message	"OK" if no error occurred, otherwise the error message will appear here.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/service/819</code>
--------------	--

```
OUTPUT {
  "info": [
    {
      "serviceId": 819,
      "serviceName": "test",
      "serviceType": 0,
      "serviceAtLeastValue": 0
      "serviceDegradedThreshold": 0,
      "raiseEvents": true,
      "treatUnknownAsDown": false,
      "serviceSlaGoal": 99.0,
      "serviceWebImage": 0,
      "descriptiveAlias": "",
      "ownerId": "admin",
      "hasAdminPermission": true,
      "availableEyeServers": null,
      "defaultEyeServerId": null,
      "users": [
        {
          "first": 3,
          "second": "admin",
        },
        {
          "first": 4,
          "second": "user",
        }
      ],
      "aggregateEnable": false
      "serviceTag": "Standard"
      "message": "ok"
    },
    "componentIds": [
      {
        "serverId": "1758b1ea-2e6e-4bb2-82db-810a153293a2"
        "objectId": "795"
      }
    ]
  ]
}
```

```
}

```

PUT Method detail

Modify the settings of the service. Note, to modify the components of a service, you will need to use the POST method below.

Request Parameters

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/service/830 -X PUT -H "content-type:application/json" -d { "serviceName": "test1" }</pre>
OUTPUT	<pre>{ "info": [{ "serviceId": 830, "serviceName": "test", "serviceType": 0, "serviceAtLeastValue": 0 "serviceDegradedThreshold": 0, "raiseEvents": true, "treatUnknownAsDown": false, "serviceSlaGoal": 99.0, "serviceWebImage": 0, "descriptiveAlias": "", "ownerId": "admin", "hasAdminPermission": true, "availableEyeServers": null, "defaultEyeServerId": null, "users": [{ "first": 3, "second": "admin",</pre>

```
        },
        {
            "first": 4,
            "second": "user",
        }
    ],
    "aggregateEnable": false
    "serviceTag": "Standard"
    "message": "ok"
},
"componentIds": [
    {
        "serverId": "1758b1ea-2e6e-4bb2-82db-810a153293a2"
        "objectId": "795"
    }
]
```

POST Method detail

Add or remove components to or from the service. You can retrieve device object keys by calling 'api/inventory'.

Request Parameters

Name	Description
addList	a list of IDs for components to be added to the service.
removeList	a list of IDS for components to be removed from the service.

Response Parameters

Returns the modified service if valid, otherwise returns the error message.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/service/825 -X POST -H "content-type:application/json" -d { "addList": [{ "serverId": "1758b1ea-2e9e-4bb2-82db-810a153293a2", "id": 795 }] }</pre>
OUTPUT	<pre>{ "info": [{ "serviceId": 825, "serviceName": "ok", "serviceType": 0, "serviceAtLeastValue": 0 "serviceDegradedThreshold": 0, "raiseEvents": true, "treatUnknownAsDown": false, "serviceSlaGoal": 99.0, "serviceWebImage": 0, "descriptiveAlias": "", "ownerId": "admin", "hasAdminPermission": true, "availableEyeServers": null, "defaultEyeServerId": null, "users": [{ "first": 3, "second": "admin", }, { "first": 4, "second": "user", }] }] }</pre>

```
        ],
        "aggregateEnable": false
        "serviceTag": "Standard"
        "message": "ok"
    },
    "componentIds": [
        {
            "serverId": "1758b1ea-2e6e-4bb2-82db-810a153293a2"
            "objectId": "795"
        }
    ]
}
```

DELETE Method detail

Deletes the service.

Response

"OK" messaged displayed when successfully deleted.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/service/830 -X DELETE</code>
OUTPUT	<pre>{ "message": "ok" }</pre>

OS Services Summary

List all OS Service rules, create a new rule or delete all rules.

URL: osService

- [Method Summary](#)
- [GET Method detail](#)
 - [Response data keys](#)
 - [Examples](#)
- [POST Method detail](#)
 - [Request Parameters](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - returns a list of all OS Services rules.
- POST Method - creates a new OS Services rule within the list.
- DELETE Method - deletes all OS Services rules.

GET Method detail

Returns a list of all OS Services rules.

Response data keys

Name	Description
id	ID number of the OS Service rule.
serviceName	the OS Service name that is being filtered for in the rule.
description	description of the OS Service, as entered by the user.
filterUsing	type of filter to apply to the service name, either: • 0: Equals • 1: Contains

operatingSystem	operating system of the rule: • 0: Windows • 1: Linux (currently inapplicable).
enabled	if the OS Service rule is currently enabled, either 'true' or 'false'.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/osService?media=json</code>
--------------	---

OUTPUT	<pre>{ "serviceRules": [{ "id": 1, "serviceName": "EOTS", "description": "Entuity", "filterUsing": 0, "operatingSystem": 0 "enabled": true }, { "id": 1, "serviceName": "wuauserv", "description": "Windows Update", "filterUsing": 0, "operatingSystem": 0 "enabled": false }, { "id": 1, "serviceName": "CiscoAMP", "description": "CiscoAMP. Version number follows name", "filterUsing": 1, "operatingSystem": 0 "enabled": true }] }</pre>
INPUT	<pre>curl -u admin:admin https://localhost/api/osService?media=xml</pre>

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <osServiceList xmlns:ns5="http://www.entuity.com/webrpc" xmlns:ns2="http:// www.entuity.com/schemas/webUI" xmlns:ns4="http://www.entuity.com/schemas /eventengine" xmlns:ns3="http://www.entuity.com/schemas/flow"> <serviceRules> <serviceRules description="Entuity" enabled="true" filterUsing="0" id="1" operatingSystem="0" serviceName="EOTS"/> <serviceRules description="Windows Update" enabled="false" filterUsing="0" id="2" operatingSystem="0" serviceName="wuauaserv"/> <serviceRules description="CiscoAMP. Version number follows name" enabled="true" filterUsing="1" id="3" operatingSystem="0" serviceName=" CiscoAMP"/> </serviceRules> </osServiceList></pre>
---------------	--

POST Method detail

Creates a new OS Services rule within the list.

Request Parameters

Name	Description
serviceName	the OS Service name that is being filtered for in the rule.
description	description of the OS Service, as entered by the user.
filterUsing	type of filter to apply to the service name, either: • 0: Equals • 1: Contains
operatingSystem	operating system of the rule: • 0: Windows • 1: Linux (currently inapplicable).
enabled	if the OS Service rule is currently enabled, either 'true' or 'false'.

Response

Lists and details all OS Services on the server, including the OS Service that has just been created.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/osService -X POST -H "content-type:application/json" -d { "serviceRules": [{ "serviceName": "TestService3", "description": "test service 3", "filterUsing": 0, "operatingSystem": 0 "enabled": true }] }</pre>
-------	---

OUTPUT	<pre>{ "serviceRules" : [{ "id" : 1, "serviceName": "TestService", "description": "test service", "filterUsing": 0, "operatingSystem": 0, "enabled": true, "auditLogWriter" : 1 }, { "id" : 2, "serviceName": "TestService2", "description": "test service 2", "filterUsing": 0, "operatingSystem": 1, "enabled": true, "auditLogWriter" : 1 }, { "id" : 3, "serviceName": "TestService3", "description": "test service 3", "filterUsing": 1, "operatingSystem": 1, "enabled": true, "auditLogWriter" : 1 }] }</pre>
INPUT	<pre>curl -u admin:admin https://localhost/api/osService?media=xml -X POST -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <osServiceList> <serviceRules> <serviceRules description="test service 3" enabled="true" filterUsing="0" operatingSystem="0" serviceName="TestService3"/> </serviceRules> </osServiceList>'</pre>

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <osServiceList xmlns:ns5="http://www.entuity.com/webrpc" xmlns:ns2="http:// www.entuity.com/schemas/webUI" xmlns:ns4="http://www.entuity.com/schemas /eventengine" xmlns:ns3="http://www.entuity.com/schemas/flow"> <serviceRules> <serviceRules description="test service" enabled="true" filterUsing="0" id="1" operatingSystem="0" serviceName="TestService"/> <serviceRules description="test service 2" enabled="true" filterUsing="0" id="2" operatingSystem="1" serviceName="TestService2"/> <serviceRules description="test service 3" enabled="true" filterUsing="1" id="3" operatingSystem="1" serviceName="TestService3"/> </serviceRules> </osServiceList></pre>
---------------	---

DELETE Method detail

Deletes all OS Services rules.

Response

"Deleted {X} OS Service Rules" message if the device was removed successfully, an error message otherwise.

Example

INPUT	curl -u admin:admin https://localhost/api/osService?media=json -X DELETE
OUTPUT	<pre>{ "message": "Deleted 2 OS Service Rules" }</pre>
INPUT	curl -u admin:admin https://localhost/api/osService?media=xml-X DELETE
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <result message="Deleted 2 OS Service Rules" xmlns:ns5="http://www. entuity.com/webrpc" xmlns:ns2="http://www.entuity.com/schemas/webUI" xmlns:ns4="http://www.entuity.com/schemas/eventengine" xmlns:ns3="http ://www.entuity.com/schemas/flow"/></pre>

OS Services Detail

List, update or delete an OS Service rule.

URL: `osService/{id}`

- [Method Summary](#)
- [GET Method detail](#)
 - [Request Parameters](#)
 - [Response data keys](#)
 - [Example](#)
- [PUT Method detail](#)
 - [Request Parameters](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - returns a single OS Service rule.
- PUT Method - update this rule with the given OS Service rule.
- DELETE Method - deletes the OS Service rule.

GET Method detail

Returns the specified OS Service rule.

Request Parameters

None.

Response data keys

Name	Description
id	ID number of the OS Service rule.
serviceName	the OS Service name that is being filtered for in the rule.
description	description of the OS Service, as entered by the user.

filterUsing	type of filter to apply to the service name, either: • 0: Equals • 1: Contains
operatingSystem	operating system of the rule: • 0: Windows • 1: Linux (currently inapplicable).
enabled	if the OS Service rule is currently enabled, either 'true' or 'false'.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/osService/1?media=json</code>
OUTPUT	<pre>{ "id": 1, "serviceName": "EOTS", "description": "Entuity", "filterUsing": 0, "operatingSystem": 0 "enabled": true }</pre>

INPUT	<code>curl -u admin:admin https://localhost/api/osService/1?media=xml</code>
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:osServiceRule description="Entuity" enabled="true" filterUsing="0" id="1" operatingSystem="0" serviceName="EOTS" xmlns:ns5="http://www. entuity.com/webrpc" xmlns:ns2="http://www.entuity.com/schemas/webUI" xmlns: ns4="http://www.entuity.com/schemas/eventengine" xmlns:ns3="http://www. entuity.com/schemas/flow"/></pre>

PUT Method detail

Updates the OS Service rule.

Request Parameters

Name	Description
------	-------------

serviceName	the OS Service name that is being filtered for in the rule.
description	description of the OS Service, as entered by the user.
filterUsing	type of filter to apply to the service name, either: • 0: Equals • 1: Contains
operatingSystem	operating system of the rule: • 0: Windows • 1: Linux (currently inapplicable).
enabled	if the OS Service rule is currently enabled, either 'true' or 'false'.

Response

"OK" if OS Service rule updated.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/osService/1 -X PUT -H "content-type:application/json" -d { "serviceName": "TestService", "description": "test service", "filterUsing": 0, "operatingSystem": 0 "enabled": true }</pre>
OUTPUT	<pre>{ "message": "OK" }</pre>

INPUT	<pre>curl -u admin:admin https://localhost/api/osService/1?media=xml -X POST -H "content-type:application/xml" -d \ '<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:osServiceRule description="test service" enabled="true" filterUsing="0" id="1" operatingSystem="0" serviceName="TestService"/>'</pre>
--------------	---

OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:osServiceRule description="test service" enabled="true" filterUsing=" 0" id="1" operatingSystem="0" serviceName="TestService" xmlns:ns5="http://w ww.entuity.com/webrpc" xmlns:ns2="http://www.entuity.com/schemas/webUI" xmlns:ns4="http://www.entuity.com/schemas/eventengine" xmlns:ns3="http://ww w.entuity.com/schemas/flow"/></pre>
---------------	--

DELETE Method detail

Deletes the OS Service rule.

Request

The ID of the rule that is to be deleted is sent as a part of a URL.

Response

“Deleted OS Service Rule with Id{id}” message if the device was removed successfully, an error message otherwise.

Example

INPUT	curl -u admin:admin https://localhost/api/osService/2?media=json -X DELETE
OUTPUT	<pre>{ "message": "Deleted OS Service Rule with Id 2" }</pre>

INPUT	curl -u admin:admin https://localhost/api/osService/2?media=xml -X DELETE
OUTPUT	<pre><?xml version="1.0" encoding="UTF-8" standalone="yes"?> <result message="Deleted OS Service Rule with Id 2" xmlns:ns5="http://www .entuity.com/webrpc" xmlns:ns2="http://www.entuity.com/schemas/webUI" xmlns:ns4="http://www.entuity.com/schemas/eventengine" xmlns:ns3="http:// www.entuity.com/schemas/flow"/></pre>

User Defined REST Pollers

List the current user defined REST pollers on the Entuity server, add new user defined REST pollers.

URL: ud/pollers

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists the current user defined REST pollers on the Entuity server.
- POST Method - create a new user defined REST poller.

GET Method details

Lists the user defined REST pollers on the Entuity server.

Response

Response includes a list of the user defined REST pollers on the Entuity server.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ud/pollers</code>
-------	---

```

OUTPUT {
  "items" : [ {
    "pollerId" : 1,
    "pollerType" : "API",
    "pollerName" : "AssocComp",
    "creationTime" : 1638371491,
    "createdBy" : "admin",
    "lastModified" : 1638371491,
    "modifiedBy" : "admin",
    "version" : 1.0,
    "revision" : 5,
    "priority" : 2002.0,
    "serverId" : null,
    "errorCount" : 0,
    "enabled" : false,
    "contextTypeName" : "PingOnlyDevice",
    "extendedTypeName" : null,
    "componentTypeName" : "udp_serviceInfo",
    "description" : "",
    "stepCount" : 1,
    "stepNames" : "AttrPolling",
    "serverIds" : null,
    "missingServerIds" : null
  } ],
  "count" : 1
}

```

POST Method details

Creates a new user defined poller.

Request

User defined poller parameters

Name	Description
------	-------------

pollerType	type of poller. Currently only "API" is supported, but "SNMP" may be supported in future versions.
pollerCollectorType	sub-type of poller. Currently only "ASSOCIATED_COMPONENT" is supported.
pollerName	unique poller name, without spaces. Alpha numeric characters and underscores are allowed but other characters are invalid. Should start with udp_ and this will be added if missing.
contextTypeName	StormWorks type name of the type of object this poller will connect to. This will be a device or associated device type such as "DeviceEx", "PingOnlyDevice" or "MerakiDevice"
componentTypeDetails	specifies details of the component type of object(s) that will be created by the poller. For ASSOCIATED_COMPONENT pollers (the only ones currently supported) the typeName field should be the same as the pollerName, and a typeDisplayName field should also be supplied to specify the display name for the single associated object.
priority	optional number determining collector priority. Can be omitted.
filterDefinition	determines which of the applicable objects the poller will actually be applied to. Can be omitted or left null to apply to all of them.
steps	list of the poller steps within this poller.
steps/authDetails	authentication method for this poller step.
steps/authDetails/authType	authentication type. One of "AUTH_NONE", "AUTH_FIXED_CREDENTIALS" or "AUTH_DEVICE_CREDENTIALS"
steps/authDetails/credential	only needed if authType is AUTH_FIXED_CREDENTIALS
steps/authDetails/credential/id	ID of the credential set to be used
steps/authDetails/credential/name	name of the credential set to be used
steps/stepName	name of the step. Must be unique within the poller and consists only of alphanumeric characters and underscores.
steps/endpointURLSample	example URL entered in the UI. Can be omitted or left blank for Rest API calls.
steps/host	host to connect to. Use "\${host}" to automatically use the host of the device the poller is running under, or a fixed string to always connect to the same host.
steps/apiPath	path part of the Rest API URL, including any query part and any interpolated values of the form \${placeholder}.
steps/methodType	HTTP request method, either "GET" or "POST".
steps/port	port number to connect to. Can be omitted for the default HTTP or HTTPS ports 80 / 443 based on the useSSL value.

steps/useSSL	either "true" or "false" depending on whether the request should be made using HTTPS or HTTP.
steps/pathToData	optional starting point for all attribute source paths in this step. Can be omitted or left blank if the full path is specified in each attribute's sourcePath.
steps /attributeMappings	information on the attributes added by this step.
steps /attributeMapping s /nameToMapping	map from attribute name to attribute definition. The key for each entry should be unique and match the value of the swAttrName in the value part. Attribute names must start with udp_ and this will be added automatically if omitted.
steps /attributeMapping s /nameToMapping /sourcePath	path to the data for this attribute within the JSON or XML returned by the API request. This can be simply the name of a property in the JSON or a more complex expression.
steps /attributeMapping s /nameToMapping /swAttrName	attribute name that will be used in the StormWorks data model. Must begin with udp_ and only contain alpha numeric characters and underscores.
steps /attributeMapping s /nameToMapping /swAttrDisplayNa me	display name used for this attribute in the UI (dashboards, reports etc.)
steps /attributeMapping s /nameToMapping /dataType	data type to store the attribute value as. Valid values include "string", "float", "integer". If omitted will default to "string".
steps /attributeMapping s /nameToMapping /displayType	optional. How the value will be displayed in the UI. A displayType can allow the data to be shown including e.g. appropriate units and formatting
steps /attributeMapping s /nameToMapping /visibility	optional. Visibility level for this attribute. Can be one of "SHORTLISTED", "VISIBLE", "ADVANCED" or "HIDDEN".
steps /attributeMapping s /nameToMapping /enumerable	optional. If true this attribute will be considered to have a limited enumeration of values. This makes it eligible for example for being used as the data to drive the groupings of a generic pie chart.

steps /attributeMapping s /nameToMapping /graphable	optional. If true this attribute will be considered a valid choice for display on a chart, if false it will not. If omitted then a default will be determined based upon the data type (numeric values are graphable, strings are not)
steps /attributeMapping s /nameToMapping /searchable	optional. If true this attribute will be searchable using the search tool
steps /attributeMapping s /nameToMapping /statusMap	optional. A mapping of value on to status category which is only relevant for attributes that are used as the status of the object they are on
steps /attributeMapping s /nameToMapping /range	optional. Defines the minimum and maximum values this attribute can take.
steps /attributeMapping s /nameToMapping /transform	optional. Provides a mapping from raw value (as extracted from the data) to displayed value. Useful for e.g. numeric data representing a set of possible states (set enumerable to true in this case also)
steps /attributeMapping s /nameToMapping /transform/name	name of transform
steps /attributeMapping s /nameToMapping /transform /displayName	display name of transform, used in the UI.
steps /attributeMapping s /nameToMapping /transform /inputType	input data type, e.g. int32, string, etc.
steps /attributeMapping s /nameToMapping /transform /outputType	output data type, usually string

steps /attributeMapping s /nameToMapping /transform /mapping	map of input values to output values
steps /attributeMapping s /nameToMapping /eventInfo	optional. Information on what if any events to raise when this attribute's value changes
steps /attributeMapping s /nameToMapping /eventInfo /eventMode	either "NONE", "THRESHOLD" or "STATUS"
steps /attributeMapping s /nameToMapping /eventInfo /mapping	map of values to event severity
steps /attributeMapping s /nameToMapping /eventThresholds	optional. List of thresholds for events on this attribute.
steps /attributeMapping s /nameToMapping /eventThresholds /name	threshold name. Must be unique.
steps /attributeMapping s /nameToMapping /eventThresholds /displayName	threshold display name
steps /attributeMapping s /nameToMapping /eventThresholds /description	optional description of this threshold

steps /attributeMapping s /nameToMapping /eventThresholds /groupName	group this threshold belongs to
steps /attributeMapping s /nameToMapping /eventThresholds /displayUnits	units for the threshold value. Can be blank if no units apply.
steps /attributeMapping s /nameToMapping /eventThresholds /minValue	minimum value this threshold can have
steps /attributeMapping s /nameToMapping /eventThresholds /maxValue	Maximum value this threshold can have
steps /attributeMapping s /nameToMapping /eventThresholds /defaultValue	default value for this threshold
steps /attributeMapping s /nameToMapping /eventThresholds /enabled	flag for whether this threshold is enabled
steps /attributeMapping s /nameToMapping /eventThresholds /userDefined	flag for whether this threshold is user defined

Response

The newly created user defined poller, as detailed GET method would return it.

Examples

Example 1: Adding attributes to an existing device

INPUT	<pre> curl -u -k admin:admin https://localhost/api/ud/pollers?pollerType=API -X POST -H "content-type:application/json" -d ' { "pollerType": "API", "pollerCollectorType": "ASSOCIATED_COMPONENT", "pollerName": "AssocComp", "contextTypeName": "PingOnlyDevice", "componentTypeDetails" : { "typeName" : "udp_AssocComp", "typeDisplayName" : "Service Info" }, "priority": 2002, "filterDefinition": null, "steps": [{ "authDetails": { "authType": "AUTH_DEVICE_CREDENTIALS" }, "stepName": "AttrPolling", "endpointURLSample": "localhost/api/service/1004?media=json", "host": "\${host}", "apiPath": "/api/service/1004?media=json", "methodType": "GET", "port": null, "useSSL": "false", "pathToData": "info", "attributeMappings": { "nameToMapping": { "udp_serviceName": { "sourcePath": "serviceName", "swAttrName": "udp_serviceName", "swAttrDisplayName": "Service Name", "dataType": "string" } } } }] } </pre>
-------	--

	<pre>] }' </pre>
OUTPUT	List of error information. List will be empty if all went well.

Example 2: Adding attributes from a public API to a custom device

INPUT	<pre> curl -u admin:admin -X POST -k "localhost/api/ud/pollers? pollerType=API&mode=BUILD_ALL_FORCE_OVERWRITE" -H "content-type: application/json" -d ' { "pollerType": "API", "pollerCollectorType": "ASSOCIATED_COMPONENT", "pollerName": "CovidGlobal", "contextTypeName": "UserCreatedNode", "componentTypeDetails": { "typeName": "udp_CovidGlobal", "typeDisplayName": "Covid Global Rates", "description": "", "visibility": "VISIBLE" }, "filterDefinition": { "filterType": "FILTER_ATTRIBUTE_EQUALS_ANY", "attrDef": "name", "attrValue": { "array": ["CovidData"] } }, "obtainRate": 86400, "keepTime": 604800, "steps": [{ "pollerStepType": "POLLING", "stepName": "Step 1", "authDetails": { </pre>
--------------	--

```
"authType": "AUTH_NONE"
},
"endpointURLSample": "https://coronavirus.m.pipedream.net",
"host": "coronavirus.m.pipedream.net",
"apiPath": "/",
"methodType": "GET",
"port": "",
"useSSL": true,
"attributeMappings": {
  "nameToMapping": {
    "udp_CovidGlobal_confirmed": {
      "swAttrName": "udp_CovidGlobal_confirmed",
      "swAttrDisplayName": "Global Confirmed Cases",
      "dataType": "uint32",
      "displayType": "string",
      "visibility": "SHORTLISTED",
      "enumerable": false,
      "searchable": false,
      "statusMap": {},
      "eventInfo": {
        "eventMode": "NONE",
        "mapping": null
      }
    },
    "sourcePath": "summaryStats.global.confirmed"
  },
  "udp_CovidGlobal_deaths": {
    "swAttrName": "udp_CovidGlobal_deaths",
    "swAttrDisplayName": "Global Covid Deaths",
    "dataType": "uint32",
    "displayType": "string",
    "visibility": "SHORTLISTED",
    "enumerable": false,
    "searchable": false,
    "statusMap": {},
    "eventInfo": {
      "eventMode": "NONE",
```

	<pre> "mapping": null }, "sourcePath": "summaryStats.global.deaths" } } }] }' </pre>
OUTPUT	List of error information. List will be empty if all went well.

Example 3: Adding sub components to a custom device

INPUT	<pre> curl -u admin:admin -X POST -k "localhost/api/ud/pollers? pollerType=API&mode=BUILD_ALL_FORCE_OVERWRITE" -H "content-type: application/json" -d ' { "pollerType": "API", "pollerCollectorType": "SUBCOMPONENTS", "pollerName": "CovidByCountry", "contextTypeName": "UserCreatedNode", "componentTypeDetails": { "typeName": "udp_CovidByCountry", "typeDisplayName": "Covid By Country", "description": "", "visibility": "VISIBLE" }, "filterDefinition": { "filterType": "FILTER_ATTRIBUTE_EQUALS_ANY", "attrDef": "name", "attrValue": { "array": ["CovidData"] } } }, </pre>
--------------	---

```
"obtainRate": 86400,
"keepTime": 604800,
"steps": [
  {
    "pollerStepType": "DISCOVERY",
    "stepName": "Step 1",
    "authDetails": {
      "authType": "AUTH_NONE"
    },
    "endpointURLSample": "https://coronavirus.m.pipedream.net/",
    "host": "coronavirus.m.pipedream.net",
    "apiPath": "/",
    "methodType": "GET",
    "port": "",
    "useSSL": true,
    "pathToData": "rawData",
    "componentTypeDetails": {
      "typeName": "udp_CountryCovidStats",
      "typeDisplayName": "Country Covid Stats",
      "description": "",
      "visibility": "VISIBLE"
    },
    "attributeMappings": {
      "uniqueKeySourcePath": "Combined_Key",
      "componentNameSourcePath": "Country_Region"
    }
  },
  {
    "pollerStepType": "POLLING",
    "stepName": "Step 2",
    "authDetails": {
      "authType": "AUTH_NONE"
    },
    "endpointURLSample": "https://coronavirus.m.pipedream.net/",
    "host": "coronavirus.m.pipedream.net",
    "apiPath": "/",
```

```
"methodType": "GET",
"port": "",
"useSSL": true,
"isDataForAllChildren": true,
"pathToData": "rawData",
"parentPath": "udp_CountryCovidStats",
"componentTypeDetails": {
  "typeName": "udp_CountryCovidStats"
},
"attributeMappings": {
  "nameToMapping": {
    "udp_CovidByCountry_Confirmed": {
      "swAttrName": "udp_CovidByCountry_Confirmed",
      "swAttrDisplayName": "Confirmed Cases",
      "dataType": "uint32",
      "displayType": "string",
      "visibility": "SHORTLISTED",
      "enumerable": false,
      "searchable": false,
      "statusMap": {},
      "eventInfo": {
        "eventMode": "NONE",
        "mapping": null
      },
    },
    "sourcePath": "rawData[1].Confirmed"
  },
  "udp_CovidByCountry_Deaths": {
    "swAttrName": "udp_CovidByCountry_Deaths",
    "swAttrDisplayName": "Covid Deaths",
    "dataType": "uint32",
    "displayType": "string",
    "visibility": "SHORTLISTED",
    "enumerable": false,
    "searchable": false,
    "statusMap": {},
    "eventInfo": {
```

	<pre> "eventMode": "NONE", "mapping": null }, "sourcePath": "rawData[5].Deaths" } }, "uniqueKeySourcePath": "Combined_Key" }] }' </pre>
OUTPUT	List of error information. List will be empty if all went well.

Example 4: Adding basic support for vCenter VM monitoring

INPUT	<pre> curl -u admin:admin -X POST -k "localhost/api/ud/pollers? pollerType=API&mode=BUILD_ALL_FORCE_OVERWRITE" -H "content-type: application/json" -d ' { "pollerType" : "API", "pollerCollectorType" : "SUBCOMPONENTS", "pollerName" : "vCenterExtras", "contextTypeName" : "VirtualizationPlatformDevice", "componentTypeDetails" : { "typeName" : "udp_vCenterExtras", "typeDisplayName" : "vCenter Extras", "description" : "", "visibility" : "VISIBLE" }, "filterDefinition" : { "filterType" : "FILTER_ATTRIBUTE_NOT_EQUALS", "attrDef" : "name", "attrValue" : "", "inverted" : false, "additionalORedFilters" : [] }, } </pre>
--------------	--

```

"obtainRate" : 300,
"keepTime" : 604800,
"steps" : [ {
  "pollerStepType" : "SETUP",
  "stepName" : "Auth",
  "authDetails" : {
    "authType" : "AUTH_DEVICE_CREDENTIALS",
    "credential" : {
      "id" : 3,
      "deviceId" : 0,
      "kind" : "SHARED"
    }
  },
  "endpointURLSample" : "https://vcsa-lon-01.entuity.local/rest/com/vmware/cis/session",
  "host" : "",
  "apiPath" : "/rest/com/vmware/cis/session",
  "methodType" : "POST",
  "port" : "",
  "useSSL" : true,
  "mediaType" : "json",
  "attributeMappings" : { },
  "body" : "",
  "variables" : [ {
    "name" : "sessionApiKey",
    "sourcePath" : "value"
  } ]
}, {
  "pollerStepType" : "DISCOVERY",
  "stepName" : "Discovery1",
  "authDetails" : {
    "authType" : "AUTH_NONE"
  },
  "endpointURLSample" : "https://vcsa-lon-01.entuity.local/rest/vcenter/vm",
  "host" : "",
  "apiPath" : "/rest/vcenter/vm",

```

```

    "methodType" : "GET",
    "port" : "",
    "useSSL" : true,
    "pathToData" : "value",
    "componentTypeDetails" : {
      "typeName" : "udp_vCenterVM",
      "typeDisplayName" : "vCenter VM",
      "description" : "",
      "visibility" : "VISIBLE"
    },
    "attributeMappings" : {
      "uniqueKeySourcePath" : "vm",
      "componentNameSourcePath" : "name"
    },
    "headers": [ {
      "name" : "vmware-api-session-id",
      "value" : "${vars::sessionApiKey}"
    } ]
  }, {
    "pollerStepType" : "POLLING",
    "stepName" : "Polling1",
    "authDetails" : {
      "authType" : "AUTH_NONE"
    },
    "endpointURLSample" : "https://vcsa-lon-01.entuity.local/rest/vcenter/vm/vm-100",
    "host" : "",
    "apiPath" : "/rest/vcenter/vm/${ctx::id}",
    "methodType" : "GET",
    "port" : "",
    "useSSL" : true,
    "isDataForAllChildren" : false,
    "parentPath" : "udp_vCenterVM",
    "componentTypeDetails" : {
      "typeName" : "udp_vCenterVM"
    },
    "attributeMappings" : {

```

```

"nameToMapping" : {
  "udp_vCenterExtras_guest_OS" : {
    "swAttrName" : "udp_vCenterExtras_guest_OS",
    "swAttrDisplayName" : "Guest OS",
    "dataType" : "string",
    "displayType" : "string",
    "visibility" : "SHORTLISTED",
    "enumerable" : false,
    "searchable" : false,
    "statusMap" : { },
    "eventInfo" : {
      "eventMode" : "NONE",
      "mapping" : null
    },
    "sourcePath" : "value.guest_OS"
  },
  "udp_vCenterExtras_power_state" : {
    "swAttrName" : "udp_vCenterExtras_power_state",
    "swAttrDisplayName" : "Power State",
    "dataType" : "string",
    "displayType" : "string",
    "visibility" : "SHORTLISTED",
    "enumerable" : false,
    "searchable" : false,
    "statusMap" : { },
    "eventInfo" : {
      "eventMode" : "NONE" ,
      "mapping" : null
    },
    "sourcePath" : "value.power_state"
  },
  "uniqueKeySourcePath" : "vm"
},
"headers" : [ {
  "name" : "vmware-api-session-id",

```

```

    "value" : "${vars::sessionApiKey}"
  } ]
}, {
  "pollerStepType" : "DISCOVERY",
  "stepName" : "Discovery2",
  "authDetails" : {
    "authType" : "AUTH_NONE"
  },
  "endpointURLSample" : "https://vcsa-lon-01.entuity.local/rest/vcenter
/vm/vm-100",
  "host" : "",
  "apiPath" : "/rest/vcenter/vm/${ctx::id}",
  "methodType" : "GET",
  "port" : "",
  "useSSL" : true,
  "pathToData" : "value.disks",
  "parentPath" : "udp_vCenterVM",
  "componentTypeDetails" : {
    "typeName" : "udp_vCenterDisk",
    "typeDisplayName": "vCenter Disk",
    "description" : "",
    "visibility" : "VISIBLE"
  },
  "attributeMappings" : {
    "uniqueKeySourcePath" : "key",
    "componentNameSourcePath" : "value.label"
  },
  "headers" : [ {
    "name" : "vmware-api-session-id",
    "value" : "${vars::sessionApiKey}"
  } ]
}, {
  "pollerStepType" : "POLLING",
  "stepName" : "Polling2",
  "authDetails" : {
    "authType" : "AUTH_NONE"
  },

```

```
"endpointURLSample" : "https://vcsa-lon-01.entuity.local/rest/vcenter
/vm/vm-100",
"host" : "",
  "apiPath" : "/rest/vcenter/vm/${ctx::id}",
  "methodType" : "GET",
"port" : "",
  "useSSL" : true,
  "isDataForAllChildren" : true,
  "pathToData" : "value.disks",
  "parentPath" : "udp_vCenterVM/udp_vCenterDisk",
  "componentTypeDetails" : {
    "typeName" : "udp_vCenterDisk"
  },
"attributeMappings" : {
  "nameToMapping" : {
    "udp_vCenterExtras_capacity" : {
      "swAttrName" : "udp_vCenterExtras_capacity",
      "swAttrDisplayName" : "Capacity",
      "dataType" : "int64",
      "displayType" : "string",
      "visibility" : "SHORTLISTED",
      "enumerable" : false,
      "searchable" : false,
      "statusMap" : { },
      "eventInfo" : {
        "eventMode" : "NONE",
        "mapping" : null
      },
    },
    "sourcePath" : "value.capacity"
  },
  "udp_vCenterExtras_type" : {
    "swAttrName" : "udp_vCenterExtras_type",
    "swAttrDisplayName" : "Type",
    "dataType" : "string",
    "displayType" : "string",
    "visibility" : "SHORTLISTED",
    "enumerable" : false,
```

```
        "searchable" : false,
        "statusMap" : { },
        "eventInfo" : {
            "eventMode" : "NONE",
            "mapping" : null
        },
        "sourcePath" : "value.type"
    }
},
"uniqueKeySourcePath" : "key"
},
"headers" : [ {
    "name" : "vmware-api-session-id",
    "value" : "${vars::sessionApiKey}"
} ]
} ]
}'
```

OUTPUT List of error information. List will be empty if all went well.

User Defined REST Poller Details

List details of, modify or delete a specified user defined REST poller.

URL: ud/pollers/[id]

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Examples](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - list details of a specified user defined REST poller.
- PUT Method - modify an existing user defined REST poller.
- DELETE Method - delete a user defined REST poller

GET Method details

Lists details about the specified user defined REST poller.

Response

Name	Description
pollerId	unique ID of this poller on this specific Entuity server.
pollerType	type of poller. Currently only "API" is supported, but "SNMP" may be supported in future versions.
pollerName	unique poller name, without spaces. Alpha numeric characters and underscores are allowed but other characters are invalid. Should start with udp_ and this will be added if missing.
creationTime	unix timestamp for when this poller was first created.
createdBy	name of Entuity user that created this poller.
lastModified	unix timestamp for when this poller was last modified.

version	version number.
revision	revision number. This is incremented each time the user edits this poller.
priority	number determining collector priority.
serverId	this field is currently redundant and will always be null.
errorCount	number of errors while creating this poller. Should be zero or null for a successful poller.
pollerDef	details of this poller's definition.
pollerType	type of poller. Currently only "API" is supported, but "SNMP" may be supported in future versions.
pollerCollector Type	sub-type of poller. Currently only "ASSOCIATED_COMPONENT" is supported.
pollerId	ID of the poller.
version	version number.
revision	revision number. This is incremented each time the user edits this poller.
priority	optional number determining collector priority. Can be omitted.
contextTypeName	StormWorks type name of the type of object this poller will connect to. This will be a device or associated device type such as "DeviceEx", "PingOnlyDevice" or "MerakiDevice".
componentTypeDetails	specifies details of the component type of object(s) that will be created by the poller. For ASSOCIATED_COMPONENT pollers (the only ones currently supported) the typeName field should be the same as the pollerName, and a typeDisplayName field should also be supplied to specify the display name for the single associated object.
componentTypeDetails / typeName	unique StormWorks type name for the component type.
componentTypeDetails / typeDisplayName	display name that will be given to the associated object of the device. It can have spaces in the name. It does not need to be unique but it is best if it is.
description	description of the association.
visibility	optional. Visibility level for this attribute. Can be one of "SHORTLISTED", "VISIBLE", "ADVANCED" or "HIDDEN".
displayNameAttr	display name used for this attribute in the UI (dashboards, reports etc).
setAsParentsChildren	flag to indicate that the objects created by this poller should count as the children of the context object in e.g. the Explorer tree. It is only available if the context type is a user defined one, so is currently redundant.
obtainRate	frequency (in seconds) that the data will be polled at. Usually restricted to the following values: 60, 300, 600, 900, 1200, 1800, 3600, 21600, 43200, 86400

keepTime	length of time (in seconds) which polled data will be retained. Usually restricted to the following values: 3600, 86400, 604800, 1209600, 3024000, 16934400
steps	list of the poller steps within this poller.
stepName	name of the step. Must be unique within the poller and consists only of alphanumeric characters and underscores.
authDetails	authentication method for this poller step.
authType	authentication type. One of "AUTH_NONE", "AUTH_FIXED_CREDENTIALS" or "AUTH_DEVICE_CREDENTIALS"
endpointURL	example URL entered in the UI. Can be omitted or left blank for REST API calls.
host	host to connect to. Use "\${host}" to automatically use the host of the device the poller is running under, or a fixed string to always connect to the same host.
apiPath	path part of the Rest API URL, including any query part and any interpolated values of the form \${ <i>placeholder</i> }.
methodType	HTTP request method, either "GET" or "POST".
port	port number to connect to. Can be omitted for the default HTTP or HTTPS ports 80 / 443 based on the useSSL value.
steps/useSSL	either "true" or "false" depending on whether the request should be made using HTTPS or HTTP.
steps /pathToData	optional starting point for all attribute source paths in this step. Can be omitted or left blank if the full path is specified in each attribute's sourcePath.
steps /attributeMappings	information on the attributes added by this step.
steps /attributeMappings /nameToMapping	a map from attribute name to attribute definition. The key for each entry should be unique and match the value of the swAttrName in the value part. Attribute names must start with udp_ and this will be added automatically if omitted.
steps /attributeMappings /nameToMapping /swAttrName	unique StormWorks attribute name for this attribute. Should be prepended with udp_ and the poller name so that it only needs to be unique within this poller
steps /attributeMappings /nameToMapping /swAttrDisplayName	display name for this attribute. Does not need to be unique but usually best if it is.

steps /attributeMappings /nameToMapping/dataType	data type to store the attribute value as. Valid values include "string", "float", "integer". If omitted will default to "string".
steps /attributeMappings /nameToMapping/displayType	optional. How the value will be displayed in the UI. A displayType can allow the data to be shown including e.g. appropriate units and formatting.
steps /attributeMappings /nameToMapping/visibility	optional. Visibility level for this attribute. Can be one of "SHORTLISTED", "VISIBLE", "ADVANCED" or "HIDDEN".
steps /attributeMappings /nameToMapping/graphable	optional. If true this attribute will be considered a valid choice for display on a chart, if false it will not. If omitted then a default will be determined based upon the data type (numeric values are graphable, strings are not).
steps /attributeMappings /nameToMapping/enumerable	optional. If true this attribute will be considered to have a limited enumeration of values. This makes it eligible for example for being used as the data to drive the groupings of a generic pie chart.
steps /attributeMappings /nameToMapping/searchable	optional. If true this attribute will be searchable using the search tool.
steps /attributeMappings /nameToMapping/statusMap	optional. A mapping of value on to status category which is only relevant for attributes that are used as the status of the object they are on.
steps /attributeMappings /nameToMapping/sourcePath	path to the data for this attribute within the JSON or XML returned by the API request. This can be simply the name of a property in the JSON or a more complex expression.
steps /attributeMappings /nameToMapping/eventInfo	optional. Information on what if any events to raise when this attribute's value changes.

steps /attributeMappings /nameToMapping/eventInfo /eventMode	either "NONE", "THRESHOLD" or "STATUS"
steps /attributeMappings /nameToMapping/eventInfo /mapping	map of values to event severity.

Examples

INPUT	curl -u admin:admin https://localhost/api/ud/pollers/1
OUTPUT	<pre>{ "pollerId" : 1, "pollerType" : "API", "pollerName" : "AssocComp", "creationTime" : 1638371491, "createdBy" : "admin", "lastModified" : 1638371491 "modifiedBy" : "admin", "version" : 1.0, "revision" : 1, "priority" : 2002.0, "serverId" : "400e61cf-873d-4504-8784-f9075a9a5150", "errorCount" : 0, "pollerDef" : { "pollerType" : "API" "pollerCollectorType" : "ASSOCIATED_COMPONENT", "pollerId" : 1, "pollerName" : "AssocComp", "version" : 1.0, "revision" : 1, "priority" : 2002.0, "contextTypeName" : "PingOnlyDevice", "componentTypeDetails" : { "typeName" : "udp_serviceInfo",</pre>

```

    "typeDisplayName" : "Service Info",
    "keyAttribute" : "uDCIndexUserDefined",
    "setAsParentsChildren" : false
  },
  "steps" : [ {
    "stepName" : "AttrPolling",
    "authDetails" : {
      "authType" : "AUTH_DEVICE_CREDENTIALS"
    },
    "endpointURLSample" : "localhost/api/service/1004?media=json",
    "host" : "${host}",
    "apiPath" : "/api/service/1004?media=json",
    "methodType" : "GET",
    "useSSL" : false,
    "pathToData" : "info",
    "attributeMappings" : {
      "nameToMapping" : {
        "udp_serviceName" : {
          "swAttrName" : "udp_serviceName",
          "swAttrDisplayName" : "Service Name",
          "dataType" : "string",
          "sourcePath" : "serviceName"
        }
      }
    }
  } ]
}

```

PUT Method details

Modifies the parameters of the specified user defined REST poller.

Request

Request has the same structure as the POST request for adding a new user defined REST poller. You need to specify only properties you want to be changed.

Response

On success returns the poller definition for the newly updated poller.

Examples

INPUT	<pre>curl -u -k admin:admin https://localhost/api/ud/pollers/1 -X PUT -H "content-type:application/json" -d ' { "pollerName": "ServiceInfoOnPingDevice" }'</pre>
OUTPUT	<pre>{ "pollerId" : 1, "pollerType" : "API", "pollerName" : "ServiceInfoOnPingDevice", "creationTime" : 1638461440, "createdBy" : "admin", "lastModified" : 1638461705, "modifiedBy" : "admin", "version" : 1.0, "revision" : 2, "priority" : 2000.0, "serverId" : "400e61cf-873d-4504-8784-f9075a9a5150", "errorCount" : 0, "pollerDef" : { "pollerType" : "API", "pollerCollectorType" : "ASSOCIATED_COMPONENT", "pollerId" : 1, "pollerName" : "ServiceInfoOnPingDevice", "version" : 1.0, "revision" : 2, "priority" : 2000.0, "contextTypeName" : "PingOnlyDevice", "componentTypeDetails" : { "typeName" : "udp_serviceInfo", "typeDisplayName" : "Service Info", "keyAttribute" : "uDCIndexUserDefined",</pre>

```
"setAsParentsChildren" : false
},
"steps" : [ {
  "stepName" : "AttrPolling",
  "authDetails" : {
    "authType" : "AUTH_DEVICE_CREDENTIALS"
  },
  "endpointURLSample" : "localhost/api/service/1004?media=json",
  "host" : "${host}",
  "apiPath" : "/api/service/1004?media=json",
  "methodType" : "GET",
  "useSSL" : false,
  "pathToData" : "info",
  "attributeMappings" : {
    "nameToMapping" : {
      "udp_serviceName" : {
        "swAttrName" : "udp_serviceName",
        "swAttrDisplayName" : "Service Name",
        "dataType" : "string",
        "sourcePath" : "serviceName"
      }
    }
  }
} ]
}
```

DELETE Method details

Deletes the selected user defined poller.

Request

No additional parameters needed.

Response

Returns a list of error information. This will be empty, i.e. just [] in the case of success.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/ud/pollers/1 -X DELETE</code>
OUTPUT	[]

Testing User Defined REST Pollers

Test a user defined REST poller.

URL: ud/test

Methods Summary

- POST Method - tests a user defined REST poller.

POST Method details

Tests a user defined REST poller.

Request

Name	Description
url	URL for the REST API to connect to.
methodType	HTTP method, usually "GET" or "POST".
credential	Credential information required for authentication.
mediaType	Specifies the media type being sent in a POST request, usually "json".
timeoutSecs	Optionally overrides the default amount of time the engine will wait before timing out the request (which is 30 seconds)
body	Defines the body text to be sent. Only required for POST requests.
headers	List of headers to be sent and their values.
variables	List of variables to be sent and their values.
prerequisites	List of prerequisite poller steps to be evaluated before this test can be executed. Each of this defines a complete poller step from a poller. Typically these would establish authentication and/or set the value of one or more variables that will be used during this test.

Response

Name	Description
returnCode	Code returned by the Entuity collection engine. One of "OK", "WARN" or "ERROR".
errorCode	Error code from the collection engine. May contain useful error information in the event of internal errors.
requestId	Internal ID used for this request by the collection engine.
type	Additional information from the collection engine.
timeStamp	Timestamp that the response was created,
lapsedTimeMs	Lapsed time in milliseconds processing the API request.
messages	List of error messages.
httpCode	HTTP code returned from the REST API endpoint.
httpErrorMessage	Empty string if HTTP request was a success, or an error string if an error occurred.
httpBody	Body of the HTTP response if successful. This will usually be a JSON or XML string, but in some cases APIs return HTML or other data.
httpHeaders	List of HTTP headers in the response, as name-value pairs.
httpCookies	List of HTTP cookies returned in the response. This have a name, a value and a list of properties.
variables	List of variables available to subsequent poller steps. This can be null, or a list of name-value pairs.

Examples

INPUT	<pre>curl -k -u admin:admin https://localhost/api/ud/test -X POST -H "content-type:application/json" -d '{ "url" : "https://localhost/api/service/1644?media=json", "credential" : { "attributes" : { "UserAndPasswordAttributes" : { "username" : "admin", "password" : "admin" } } } }'</pre>
--------------	---

```

OUTPUT {
  "returnCode" : "OK",
  "errorCode" : null,
  "requestId" : "DISC_1",
  "type" : null,
  "timeStamp" : 1638462971764,
  "lapsedTimeMs" : 109,
  "messages" : [ ],
  "httpCode" : 200,
  "httpErrorMessage" : "",
  "httpBody" : "{\r\n  \"info\" : {\r\n    \"serviceId\" : 1644,\r\n    \"serviceType\" : 1,\r\n    \"serviceAtLeastValue\" : 0,\r\n    \"serviceDegradedThreshold\" : 0,\r\n    \"raiseEvents\" : true,\r\n    \"treatUnknownAsDown\" : false,\r\n    \"serviceSlaGoal\" : 0.0,\r\n    \"serviceWebImage\" : 0,\r\n    \"descriptiveAlias\" : \"\", \r\n    \"ownerId\" : \"kim1\", \r\n    \"hasAdminP\" : true,\r\n    \"availableEyeServers\" : null,\r\n    \"defaultEyeServerId\" : null,\r\n    \"users\" : [ {\r\n      \"first\" : 3,\r\n      \"second\" : \"admin\", \r\n    }, {\r\n      \"first\" : 4,\r\n      \"second\" : \"user\", \r\n    }, {\r\n      \"first\" : 6,\r\n      \"second\" : \"kim1\", \r\n    }, {\r\n      \"first\" : 7,\r\n      \"second\" : \"poller1\", \r\n    } ], \r\n    \"aggregateEnable\" : false,\r\n    \"serviceTag\" : \"Standard\", \r\n    \"message\" : \"ok\", \r\n  }, \r\n  \"componentIds\" : [ {\r\n    \"serverId\" : \"400e61cf-873d-4504-8784-f9075a9a5150\", \r\n    \"objectId\" : 869 \r\n  } ] \r\n}",
  "httpHeaders" : null
}

```

Custom Webhook Groups

List and create custom webhook groups

URL: `http(s)://{server hostname}/api/webhooks/groups`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Request Parameters](#)
 - [Response data](#)
 - [Examples](#)

Methods Summary

- GET Method - list custom webhook groups
- POST Method - create a custom webhook group

GET Method details

Returns a list of custom webhook groups.

Response

Name	Description
groupID	ID of the group.
groupName	group name.
authType	type of authentication, choose from: 1 = basic 3 = API Key Header 4 = API Key Body 5 = API Key Query
authKey	name of the authentication key. This field is not applicable if the Basic authentication type is selected.

endpointCount	number of webhook endpoints assigned to this webhook group.
ruleCount	number of webhook rules associated with this webhook group.
count	total number of webhook groups on this server.

Example

INPUT	<code>curl -v -u -k admin:admin https://localhost/api/webhooks/groups?media=json -H "content-type:application/json"</code>
OUTPUT	<pre>{ "items" : [{ "groupID" : 23, "groupName" : "Junipermist", "authMethod" : { "authType" : 1, "authKey" : "not used" }, "endpointCount" : 0, "ruleCount" : 0 }, { "groupID" : 24, "groupName" : "Anewgroup", "authMethod" : { "authType" : 1, "authKey" : "not used" }, "endpointCount" : 0, "ruleCount" : 0 },], "count" : 2 }</pre>

POST Method details

Create a new custom webhook group.

Request Parameters

Name	Description
group name	name of the webhook group. Entuity recommends that the name reflects the devices in the group, because no checks are undertaken to validate the devices' manufacturer(s). You will not be able to use the same name as a group that already exists.
authMethod	authentication method.
authType	type of authentication, choose from: 1 = basic 3 = API Key Header 4 = API Key Body 5 = API Key Query
authKey	name of the authentication key. This is not applicable if the 'Basic' authentication type is selected.

Response data

A confirmation of the addition.

Examples

INPUT	<pre>curl -v -u -k admin:admin https://localhost/api/webhooks/groups?media=json -X POST -H "content-type:application/json" -d \ '{' "groupName" : "Junipermist", "authMethod" : { "authType" : "1", "authKey" : "not used" } }</pre>
--------------	--

OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0, "message" : "Webhook group 'Junipermist' created successfully with ID 23. "otherResults" : []</pre>
---------------	---

Custom Webhook Group Details

Inspect or delete a custom webhook group

URL: `http(s)://{server hostname}/api/webhooks/groups/{group name}`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [PUT Method details](#)
 - [Response](#)
 - [Examples](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - inspect a custom webhook group
- PUT Method - modify a custom webhook group
- DELETE Method - delete a custom webhook group

GET Method details

Returns information about the specified custom webhook group.

Response

Information about the custom webhook group in the following format:

Name	Description
groupID	ID of the group.
groupName	group name.

authType	type of authentication, either: type of authentication, choose from: 1 = basic 3 = API Key Header 4 = API Key Body 5 = API Key Query
authKey	name of the authentication key. This field is not applicable if the Basic authentication type is selected.
endpointCount	number of webhook endpoints assigned to this webhook group.
ruleCount	number of webhook rules associated with this webhook group.

Example

INPUT	<code>curl -v -u -k admin:admin https://localhost/api/webhooks/groups/anewgroup?media=json -H "content-type:application/json"</code>
OUTPUT	<pre>{ "groupID" : 24, "groupName" : "Anewgroup", "authMethod" : { "authType" : 1, "authKey" : "not used" }, "endpointCount" : 0, "ruleCount" : 0 }</pre>

PUT Method details

Modifies the specified custom webhook group.

Response

Name	Description
groupID	ID of the group.
groupName	group name.

authType	type of authentication, either: type of authentication, choose from: 1 = basic 3 = API Key Header 4 = API Key Body 5 = API Key Query
authKey	name of the authentication key. This field is not applicable if the Basic authentication type is selected.

Examples

INPUT	<pre>curl -u admin:admin -k https://localhost/api/webhooks/groups/anewgroup -H 'content-type:application/json' -X PUT { "groupID" : "24", "groupName" : "Anewgroup", "authMethod" : { "authType" : 1, "authKey" : "not used" }, }</pre>
OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0, "message" : "Succeeded to edit Webhook group 'anewgroup'.", "otherResults" : [] }</pre>

DELETE Method details

Deletes the specified custom webhook group.

Response

Confirmation that the group has been deleted.

Examples

INPUT	<code>curl -u admin:admin -k https://localhost/api/webhooks/groups/anewgroup -H 'content-type:application/json' -X DELETE</code>
OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0 "message" : "Groups deleted successfully.", "otherResults" : [] }</pre>

Custom Webhook Rules

List and create custom webhook rules

URL: `http(s)://{server hostname}/api/webhooks/rules`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Request Parameters](#)
 - [Examples](#)

Methods Summary

- GET Method - list custom webhook rules
- POST Method - create a custom webhook rule

GET Method details

Returns a list of custom webhook rules.

Response

Same parameter fields as used below in the POST call to create a webhook rule, plus an allocated ruleID.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/rules -H "content-type: application/json"</code>
-------	---

```

OUTPUT {
  "items" : [ {
    "ruleID" : 11,
    "ruleName" : "Meraki Device Up Rule",
    "groupID" : 23,
    "conditions" : [ {
      "eventConditionID" : 9,
      "fieldToTest" : "${deviceName}",
      "valueToMatch" : "My AP.*",
      "operator" : "LIKE",
      "transformType" : "STRING",
      "result" : false
    } ],
    "eventMapping" : {
      "eventMappingID" : 8,
      "eventSource" : "${deviceName} on ${networkName} ",
      "eventDetails" : "${deviceModel} ",
      "webhookEvent" : {
        "eventName" : "Meraki Device Came Up",
        "eventDescription" : "Test Meraki Device Came Up Event",
        "eventID" : 8,
        "eventSeverity" : 6
      },
      "eventName" : "Meraki Device Came Up",
      "eventID" : 8
    },
    "enabled" : 1,
    "payloadID" : 15
  } ],
  "count" : 1
}

```

POST Method details

Create a new custom webhook rule.

Request Parameters

Name	Description
ruleName	name of the rule.
groupID	id of the webhook group to which this rule will apply.
conditions	set of conditions that determine whether the webhook payload meets the requirements for raising a webhook event.
fieldToTest	key from which the value will be retrieved.
valueToMatch	value to compare to the value from the payload.
operator	operation that the value will use.
transform Type	type into which the value will be converted, from the following: • Int32 • Int64 • Float • Double • String
eventMapping	set of parameters by which to map the webhook event.
eventSource	informational entry of the event source.
eventDetails	informational entry of the event details
eventName	name of the custom webhook event.
eventDescriptor	description of the event. This is informational only.
eventSeverity	severity of the event, select from: 2 - Info 4 - Minor 6 - Major 8 - Severe 10 - Critical
attachPayload	if the webhook payload is to be attached to the generated event as a custom attribute (called 'webhookPayload'), either "true" or "false".

Examples

INPUT	<pre>curl -v -u -k admin:admin https://localhost/api/webhooks/rules -X POST -H "content-type:application/json" -d\ '{ "ruleName" : "Meraki Device Reboot", "groupID" : 1, "conditions" : [{ "fieldToTest" : "alertType", "valueToMatch" : ".*came up", "operator" : "LIKE", "transformType" : "STRING" }], "eventMapping" : { "eventSource" : "\${deviceName} on \${networkName}", "eventDetails" : "\${deviceName} down", "webhookEvent" : { "eventName" : "Meraki Device Reboot", "eventDescr" : "Test", "eventSeverity" : "8" } } "attachPayload" : true }</pre>
OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0, "message" : "Webhook rule 'Meraki Device Reboot' created successfully.", "otherResults" : [] }</pre>

Custom Webhook Rule Details

Inspect or delete a custom webhook rule

URL: `http(s)://{server hostname}/api/webhooks/rules/{rule id}`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - inspect a custom webhook rule
- DELETE Method - delete a custom webhook rule

GET Method details

Returns information about the specified custom webhook rule.

Response

Information about the custom webhook rule in the following format:

Name	Description
ruleName	name of the rule.
groupID	id of the webhook group to which this rule will apply.
conditions	set of conditions that determine whether the webhook payload meets the requirements for raising a webhook event.
fieldToTest	key from which the value will be retrieved.
valueToMatch	value to compare to the value from the payload.
operator	operation that the value will use.

transformType	type into which the value will be converted, from the following: • Int32 • Int64 • Float • Double • String
eventMapping	set of parameters by which to map the webhook event.
eventSource	informational entry of the event source.
eventDetails	informational entry of the event details
eventName	name of the custom webhook event.
eventDescr	description of the event. This is informational only.
eventSeverity	severity of the event, select from: 2 - Info 4 - Minor 6 - Major 8 - Severe 10 - Critical

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/rules/11 -H "content-type:application/json"</code>
--------------	---

```
OUTPUT {
  "ruleID" : 11,
  "ruleName" : "Meraki Device Up Rule",
  "groupID" : 23,
  "conditions" : [ {
    "eventConditionID" : 9,
    "fieldToTest" : "${deviceName}",
    "valueToMatch" : "My AP.*",
    "operator" : "LIKE",
    "transformType" : "STRING",
    "result" : false
  } ],
  "eventMapping" : {
    "eventMappingID" : 8,
    "eventSource" : "${deviceName} on ${networkName} ",
    "eventDetails" : "${deviceModel} ",
    "webhookEvent" : {
      "eventName" : "Meraki Device Came Up",
      "eventDescription" : "Test Meraki Device Came Up Event",
      "eventID" : 8,
      "eventSeverity" : 6
    },
    "eventName" : "Meraki Device Came Up",
    "eventID" : 8
  },
  "enabled" : 1,
  "payloadID" : 15
}
```

DELETE Method details

Deletes the specified custom webhook rule using its ruleID.

Response

Indicates success or failure of deletion.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/rules/13 -X DELETE -H "content-type:application/json"</code>
OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0, "message" : "Successfully deleted 1 rules.", "otherResults" : [] }</pre>

Custom Webhook Endpoints

List all custom webhook endpoints

URL: `http(s)://{server hostname}/api/webhooks/endpoints`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - list all custom webhook endpoints

GET Method details

Returns a list of custom webhook endpoints.

Response

Name	Description
endpointID	ID of the endpoint.
objectID	ID of the object on which the endpoint URL is found.
serverID	ID of the server on which the endpoint object is found.
payloadMax	maximum number of payloads that Entuity will store at any one time. For example, if you specify a payload maximum of 3, then the 3 most recent payloads will be stored. When a new payload is received, the oldest of the 3 saved would then be deleted and replaced with the new one. Note, the payload maximum cannot be edited if payload collection is disabled.
saveEnabled	whether payload collection is enabled, either true or false . If false, this will prevent the saving of any further payloads.
endpointName	name of the object on which the endpoint URL is found.
groupName	name of the webhook group in which the endpoint URL is found.
payloadCount	number of saved payloads.

secret	secret key, if applicable.
---------------	----------------------------

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/endpoints -H "content-type:application/json"</code>
OUTPUT	<pre>{ "items" : [{ "endpointID" : 3, "objectID" : 865, "serverID" : "{serverid}", "payloadMax" : 1, "saveEnabled" : true, "endpointName" : "bottom2960", "groupName" : "Anewgroup", "payloadCount" : 0, "secret" : "" }, { "endpointID" : 4, "objectID" : 1147, "serverID" : "{serverid}", "payloadMax" : 1, "saveEnabled" : true, "endpointName" : "quidway", "groupName" : "Anewgroup", "payloadCount" : 0, "secret" : "" }], "count" : 2 }</pre>

Custom Webhook Endpoint Details by Group

List all custom webhook endpoints for a specified webhook group

URL: `http(s)://{server hostname}/api/webhooks/groups/{group name}/endpoints`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - list all custom webhook endpoints for a specified webhook group.

GET Method details

Returns a list of the custom webhook endpoints in the specified webhook group.

Response

Name	Description
containerID	ID of the webhook group.
containerName	name of the webhook group.
endpointID	ID of the endpoint.
objectID	ID of the object on which the endpoint URL is found.
serverID	ID of the server on which the webhook group is hosted.
payloadMax	maximum number of payloads that Entuity will store at any one time. For example, if you specify a payload maximum of 3, then the 3 most recent payloads will be stored. When a new payload is received, the oldest of the 3 saved would then be deleted and replaced with the new one. Note, the payload maximum cannot be edited if payload collection is disabled.
saveEnabled	whether payload collection is enabled, either true or false . If false, this will prevent the saving of any further payloads.
endpointName	name of the object on which the endpoint URL is found.

group Name	name of the webhook group in which the endpoint URL is found.
payloadCount	number of saved payloads.
secret	secret key, if applicable.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/groups/anewgroup/endpoints -H "content-type:application/json"</code>
--------------	---

```
OUTPUT {
  "items" : [
    {
      "containerID" : 24,
      "containerName" : null,
      "endpoints" : [
        {
          "endpointID" : 3,
          "objectID" : 865,
          "serverID" : "{serverID}",
          "payloadMax" : 1,
          "saveEnabled" : true,
          "endpointName" : "bottom2960",
          "groupName" : "Anewgroup",
          "payloadCount" : 0,
          "secret" : ""
        },
        {
          "endpointID" : 4,
          "objectID" : 1147,
          "serverID" : "{serverID}",
          "payloadMax" : 1,
          "saveEnabled" : true,
          "endpointName" : "quidway",
          "groupName" : "Anewgroup",
          "payloadCount" : 0,
          "secret" : ""
        }
      ]
    }
  ],
  "count" : 2
}
```

Custom Webhook Endpoint Details

List all custom webhook endpoints

URL: `http(s)://{server hostname}/api/webhooks/endpoints/{endpoint ID}`

- [Methods Summary](#)
- [PUT Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- PUT Method - modify the specified custom webhook endpoint

PUT Method details

Modifies the specified custom webhook endpoint.

Response

Name	Description
endpointID	ID of the endpoint.
objectID	ID of the object on which the endpoint URL is found.
serverID	ID of the server on which the endpoint object is found.
payloadMax	maximum number of payloads that Entuity will store at any one time. For example, if you specify a payload maximum of 3, then the 3 most recent payloads will be stored. When a new payload is received, the oldest of the 3 saved would then be deleted and replaced with the new one. Note, the payload maximum cannot be edited if payload collection is disabled.
saveEnabled	whether payload collection is enabled, either true or false . If false, this will prevent the saving of any further payloads.

Example

INPUT	<pre>curl -k -u admin:admin https://localhost/api/webhooks/endpoints/5 -H "content-type:application/json" -X PUT '{ "endpointID" : 5, "objectID" : 854, "serverID" : "49cb73f5-4aa5-4dbf-9219-9f86a5e8fe8e", "payloadMax" : 2, "saveEnabled" : true }'</pre>
OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0, "message" : "Succeeded to edit Webhook endpoint '5'.", "otherResults" : [] }</pre>

Custom Webhook Events

List all custom webhook events

URL: `http(s)://{server hostname}/api/webhooks/events`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - list all custom webhook events.

GET Method details

Returns a list of all custom webhook events.

Response

Name	Description
eventName	name of the custom webhook event.
eventDescription	description of the event. This is informational only.
eventID	ID of the event.
eventSeverity	severity of the event, select from: 2 - Info 4 - Minor 6 - Major 8 - Severe 10 - Critical
count	total number of events.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/events -H "content-type:application/json"</code>
OUTPUT	<pre>{ "items" : [{ "eventName" : "Meraki Device Came Up", "eventDescription" : "Test Meraki Device Came Up Event", "eventID" : 8, "eventSeverity" : 6 }, { "eventName" : "Meraki Test Event", "eventDescription" : "Test Meraki Event Description", "eventID" : 10, "eventSeverity" : 10 }], "count" : 2 }</pre>

Custom Webhook Event Details

Modify a specified custom webhook events

URL: `http(s)://{server hostname}/api/webhooks/events/{webhook ID}`

- [Methods Summary](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- PUT Method - modify a specified custom webhook event.

PUT Method details

Modifies the specified custom webhook event.

Request

Name	Description
eventName	name of the custom webhook event.
eventDescription	description of the event. This is informational only.
eventID	ID of the event.
eventSeverity	severity of the event, select from: 2 - Info 4 - Minor 6 - Major 8 - Severe 10 - Critical

Response

Returns a Maintenance Settings object. See Maintenance Settings.

Examples

INPUT	<pre>curl -k -u admin:admin https://localhost/api/webhooks/events/17 -X PUT -H "content-type:application/json" -d \ { "eventName" : "Test Event Name", "eventDescription" : "Test Event Description", "eventID" : 17, "eventSeverity" : 6 }'</pre>
OUTPUT	<pre>{ "successCount" : 1, "failureCount" : 0, "errorCode" : 0, "message" : "Succeeded to edit Webhook event '17'.", "otherResults" : [] }</pre>

Custom Webhook Event Details by Group

Inspect information about events for a specified custom webhook group, and add a webhook endpoint to a specified custom webhook group.

URL: `http(s)://{server hostname}/api/webhooks/groups/{group id}/events`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Parameters](#)
 - [Example](#)

Methods Summary

- GET Method - details information about events for a specified custom webhook group, specified by group id.
- POST Method - add a webhook endpoint to a specified custom webhook group.

GET Method details

Returns information about events in the specified webhook group, specified by group id.

Response

Name	Description
eventName	ID of the
eventName	name of the custom webhook event.
eventDescription	description of the event. This is informational only.
eventID	ID of the event.

eventSeverity	severity of the event, select from: 2 - Info 4 - Minor 6 - Major 8 - Severe 10 - Critical
count	total number of events.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/groups/23/events -H "content-type:application/json"</code>
OUTPUT	<pre>{ "items" : [{ "containerID" : 23, "events" : [{ "eventName" : "Meraki Device Came Up", "eventDescription" : "Test Meraki Device Came Up Event", "eventID" : 8, "eventSeverity" : 6 }, { "eventName" : "Meraki Test Event", "eventDescription" : "Test Meraki Event Description", "eventID" : 10, "eventSeverity" : 10 }] }], "count" : 1 }</pre>

POST Method details

Add a webhook endpoint to a specified custom webhook group.

Parameters

Name	Description
objectID	ID of the object on which the endpoint URL is found.
serverID	ID of the server on which the webhook group is hosted.
payloadMax	maximum number of payloads that Entuity will store at any one time. For example, if you specify a payload maximum of 3, then the 3 most recent payloads will be stored. When a new payload is received, the oldest of the 3 saved would then be deleted and replaced with the new one. Note, the payload maximum cannot be edited if payload collection is disabled.
saveEnabled	whether payload collection is enabled, either true or false . If false, this will prevent the saving of any further payloads.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/groups/anewgroup/endpoints -X POST -H "content-type:application/json" -d \</code>
OUTPUT	<code>{ "objectID": "5482", "serverID": "{serverid}", "payloadMax": "10", "saveEnabled" : "false" }</code>

List Custom Webhook Payloads

List custom webhook payloads

URL: `http(s)://{server hostname}/api/webhooks/payloads`

- [Methods Summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - list custom webhook payloads

GET Method details

Returns a list of custom webhook payloads.

Response

Name	Description
payload	data of the payload.
endpointID	ID of the custom webhook endpoint from which the payload was received.
endpointName	name of the custom webhook endpoint from which the payload was received.
objectID	stormworks identifier for hte managed device from which this webhook was sent.
mediaType	payload media type (json).
timestamp	timestamp of receipt of the payload.
groupName	name of the custom webhook group on which the endpoint is found.
payloadID	ID of the individual payload received.
count	total number of custom webhook payloads.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/webhooks/payloads -H 'content-type:application/json'</code>
-------	---

```

OUTPUT {
  "items" : [ [ {
    "payload" : "{\n \"version\": \"0.1\", \n \"sharedSecret\": \"ce2b9bf1-
c138-473e-b8ee-0ba7f4cbd2d7\", \n \"sentAt\": \"2021-05-11T08:09:00.046974
Z\", \n \"organizationId\": \"2930418\", \n \"organizationName\": \"My
organization\", \n \"organizationUrl\": \"https://dashboard.meraki.com/o
/VjjsAd/manage/organization/overview\", \n \"networkId\": \"N_24329156\", \n
\"networkName\": \"Main Office\", \n \"networkUrl\": \"https://n1.meraki.com
//n//manage/nodes/list\", \n \"networkTags\": [\n \"region-1\", \n \"
corporate\" \n ], \n \"deviceSerial\": \"Q234-ABCD-5678\", \n \"deviceMac\":
\"e0:55:3d:10:56:8a\", \n \"deviceName\": \"My AP\", \n \"deviceUrl\": \"
https://n1.meraki.com//n//manage/nodes/new_list/000000000000\", \n \"
deviceTags\": [\n \"tag1\", \n \"tag2\" \n ], \n \"device/Model\": \"MR34\",
\n \"alertId\": \"0000000000000000\", \n \"alertType\": {\n \"alert Type
Name\": \"JuniperMist device came up\", \n \"alertTypeId\": \"
started_reporting\" \n }, \n \"alertLevel\": \"informational\", \n \"
occurredAt\": \"2018-02-11T00:00:00.123450Z\", \n \"junk1\": \"junk1\", \n \"
junk2\": \"junk2\", \n \"alertData\": {\n \"DataThing1\": \"thing!\" \n }
\n}",
    "endpointID" : 3,
    "endpointName" : "MerakiInHouse",
    "objectID" : 818,
    "mediaType" : "json",
    "timestamp" : 1649088382,
    "groupName" : "Meraki",
    "payloadID" : 15
  } ] ],
  "count" : 1
}

```

Create Custom Webhook Payloads

Create custom webhook payloads

URL: `http(s)://{server hostname}/api/webhook/cloudAlert/Generic?serverId={serverId}&group={groupId}&objectId={objectId}`

- [Methods Summary](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- POST Method - create custom webhook payloads

POST Method details

Creates a custom webhook payloads.

Request

Once an endpoint has been created, the URL can be copied from the URL column of the Webhook Endpoints table on the Webhooks UI page.

- `serverId` = ID of the Entuity server.
- `groupId` = custom webhook group name with which the payload is associated.
- `objectId` = Entuity server StormWorks objectId from which the payload is sent from.

Response

If successful, the output is empty.

Example

INPUT	
	<pre>curl -u admin:admin -k https://localhost/api/webhook/cloudAlert/Generic?group=Junipermist&objectId=808&serverId=8d3f62ec-407b-440e-bc4f-97b8633c7fd6 -H 'content-type:application/json' -X POST -d \</pre> <pre>{</pre> <pre> "version": "0.1",</pre> <pre> "sharedSecret": "bd5f88b2-b082-45a7-8cd6-4a9440855726",</pre> <pre> "sentAt": "2021-05-11T08:09:00.046974Z",</pre> <pre> "organizationId": "2930418",</pre>

```

    "organizationName": "My organization",
    "organizationUrl": "https://dashboard.meraki.com/o/VjjsAd/manage/organization/overview",
    "networkId": "N_24329156",
    "networkName": "Main Office",
    "networkUrl": "https://n1.meraki.com//n//manage/nodes/list",
    "networkTags": [
    "region-1",
    "corporate"
    ],
    "deviceSerial": "Q234-ABCD-5678",
    "deviceMac": "e0:55:3d:10:56:8a",
    "deviceName": "My AP",
    "deviceUrl": "https://n1.meraki.com//n//manage/nodes/new_list/000000000000",
    "deviceTags": [
    "tag1",
    "tag2"
    ],
    "deviceModel": "MR34",
    "alertId": "0000000000000000",
    "alertType": {
    "alertTypeName": "JuniperMist device came up",
    "alertTypeId": "started_reporting"
    },
    "alertLevel": "informational",
    "occurredAt": "2018-02-11T00:00:00.123450Z",
    "junk1": "junk1",
    "junk2": "junk2",
    "alertData": {
    "DataThing1": "thing!"
    }
  }
}

```

OUTPUT []

License Usage

List used and available license metrics.

Lists used and available license metrics.

URL: `http(s)://{server hostname}/api/licenseinfo`

- [Method Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- GET Method - lists used and available license metrics.

GET Method detail

Returns used and available license metrics for Full, Basic, Associated, Config

Response

Response details license information and metrics.

Examples

INPUT	<code>curl -u admin:admin https://localhost/api/licenseinfo?media=json</code>
-------	---

OUTPUT	<pre> { "serverId" : "4bf1a9d8-9d65-4d9d-b2b1-ac42c0c0ad39", "expiryDate" : "Fri Sep 23 17:36:18 BST 2022", "licenseType" : "Evaluation", "platform" : "Windows", "licenses" : [{ "name" : "EOSprovost", "displayName" : "Entuity Core", "enabled" : true, "module" : false, "fromLicenseServer" : false, "usageInfo" : null, "expiryInfo" : { "licensed" : true, "expired" : false, "expiry" : 1663950978000, "asString" : "Valid: 17:36:18, 23 Sep 2022" }, "source" : "Local License File" }, { "name" : "EOSBasicDevices", "displayName" : "Basic Managed Devices", "enabled" : true, "module" : false, "fromLicenseServer" : false, "usageInfo" : { "name" : "EOSBasicDevices", "displayName" : "Basic Managed Device", "total" : 100, "used" : 0, "unlimited" : false, "available" : 100 } }, </pre>
---------------	---

IPAM - Settings and Scanning

List settings and status of IPAM, modify the settings, and manually start a scan of IPAM networks.

URL: ipam

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Response](#)
 - [Example](#)

Methods summary

- GET Method - returns settings and status of IPAM.
- PUT Method - modifies IPAM settings.
- POST Method - manually starts a scan of IPAM networks.

GET Method details

Returns information about IPAM settings and status.

Response

User info structure with the following format:

Name	Description
historyExpiry	number of days before history entries are removed from the IP History table. Must be an integer between 1 and 14, by default 7.
usageEvents	if usage % events are enabled, either 'true' or 'false'.
usageMedium	% threshold above which the network usage bar in the Networks table will turn yellow. Must be between 1 and 100, by default 50.
scanPeriodHours	number of hours between scheduled Ping, Reverse DNS, and ARP scans. Must be an integer between 1 and 24, by default 2.

usageHigh	% threshold above which the network usage bar in the Networks table will turn yellow. Must be between 1 and 100.
maxScannableLength	maximum number of IP addresses in a range that a single network can cover, if scanning is enabled, one of the following: • 0: 64 (/26) • 1: 128 (/25) • 2: 256 (/24) • 3: 512 (/23) • 4: 1024 (/22) • 5: 2048 (/21) • 6: 4096 (/20) • 7: 8192 (/19) • 8: 16384 (/18) • 9: 32768 (/17) • 10: 65536 (/16)
dhcpConflictEvents	if DHCP Conflict Events are enabled, either 'true' or 'false'.
dhcpScanPeriodHours	number of hours between scheduled DHCP server scans. Must be an integer between 2 and 48, by default 4.
activeTimeframeHours	imeframe (in hours) in which a successful scan of an IP address should fall to be considered active, by default 12. I.e., if an IP address was last successfully scanned within the last 12 hours, then the IP address is considered active. Must be an integer between 2 and 72, and must be greater than both scanPeriodHours and dchpScanPeriodHours.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam?media=json</code>
--------------	--

```
OUTPUT [
  [ "Store: OK", "Service: OK", "Scanner: No scans running", "Scheduler:
Running - Network Scan in 01:56:49, DHCP Scan in 03:56:49, Expire clean in
01:26:49" ],
  {
    "historyExpiry" : 7,
    "usageEvents" : true,
    "usageMedium" : 50,
    "scanPeriodHours" : 2,
    "usageHigh" : 80,
    "maxScannableLength" : 6,
    "dhcpConflictEvents" : true,
    "dhcpScanPeriodHours" : 4,
    "activeTimeframeHours" : 12
  }
]
```

PUT Method details

Modifies IPAM settings.

Request

Same as the GET call table above.

Response

Updated IPAM settings.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam?media=json -X PUT -H "content-type:application/json" -d \ ' { "historyExpiry" : 10, "usageEvents" : true, "usageMedium" : 50, "scanPeriodHours" : 2, "usageHigh" : 80, "maxScannableLength" : 6, "dhcpConflictEvents" : true, "dhcpScanPeriodHours" : 4, "activeTimeframeHours" : 12 } '</pre>
OUTPUT	<pre>[["Store: OK", "Service: OK", "Scanner: No scans running", "Scheduler: Running - Network Scan in 01:56:49, DHCP Scan in 03:56:49, Expire clean in 01:26:49"], { "historyExpiry" : 10, "usageEvents" : true, "usageMedium" : 50, "scanPeriodHours" : 2, "usageHigh" : 80, "maxScannableLength" : 6, "dhcpConflictEvents" : true, "dhcpScanPeriodHours" : 4, "activeTimeframeHours" : 12 }]</pre>

POST Method details

Manually starts a scan of IPAM networks.

Response

'OK' if successful.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam?media=json -X POST -H "content-type:application/json"</code>
OUTPUT	<pre>{ "message" : "OK" }</pre>

IPAM - Networks

List IPAM networks, and create a new network.

URL: ipam/network

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods summary

- GET Method - returns a list of IPAM networks.
- POST Method - creates a new network.

GET Method details

Returns a list of IPAM networks.

Response

User info structure with the following format:

Name	Description
id	ID of the network.
name	user-specified name of the network.
ipRange	IP range of the network.
role	role of the network, one of the following: • Supernet • Subnet • Range

zoneId	zone ID if applicable. Note: • a network can only belong to one zone. • multiple networks with the same IP range can exist within different zones. Consequently, the same IP addresses scanned with different zones will then result as different IP addresses, and subsequently can only be accessed from the corresponding zone/network. • ARP scans will inherit the zone from its routers.
description	user-specified description of the network.
dhcpServer	name of the DHCP server assigned to this network, if applicable.
scans	which scans are being used for this network, if any.
parentId	ID of the parent network, if applicable
usageMedium	% threshold value in which the usage will be yellow, by default a global value.
usageHigh	% threshold value in which the usage will be red, by default a global value.
skipNetwork	if the network is to be skipped for scanning ('don't scan network/address broadcast' in the UI), either 'true' or 'false'.
activeIps	number of scanned IPs that are active.
totalIps	total number of IPs covered by the network range.
conflict	names of other scopes that conflict with this one.
conflictTimestamp	timestamp of the conflict.
children	number of children of the network, if applicable.
fullPath	full path of IP range hierarchy to this network, useful for when networks have the same name but different parents, e.g. in the case of two networks both called 'test': parent 1/test parent 2/test
exceedsUsage	if the activeIps % exceeds the 'usageHigh' threshold.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/network?media=json</code>
--------------	--

OUTPUT	<pre>[{ "id" : 1, "name" : "test", "ipRange" : "10.44.2.0/24", "role" : 0, "zoneId" : 0, "description" : "", "dhcpServer" : 0, "scans" : 3, "parentId" : 0, "usageMedium" : 0, "usageHigh" : 0, "skipNetwork" : true, "activeIps" : 124, "totalIps" : 256, "conflict" : null, "conflictTimestamp" : 0, "children" : [], "fullPath" : "test", "exceedsUsage" : false, }</pre>
---------------	--

POST Method details

Create a new network.

Request

See GET call table above.

Response

'OK' if successful.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/network?media=json -X POST -H "content-type:application/json" -d \ '{ "name" : "test2", "ipRange" : "10.44.3.0/24", "role" : 0, "zoneID" : 0, "description" : "", "dhcpServer" : 0, "scans" : 0, "parentID" : 0, "usageMedium" : 0, "usageHigh" : 0, "skipNetwork" : true }'</pre>
OUTPUT	<pre>{ "message" : "OK" }</pre>

IPAM - Network Details

List details of a specified network, modify the settings of a specified network, and delete a specified network.

URL: ipam/network/{id}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Example](#)

Methods summary

- GET Method - returns details of a specified network.
- PUT Method - modifies a specified network.
- DELETE Method - delete a specified network.

GET Method details

Returns details about a specified network.

Response

User info structure with the following format:

Name	Description
id	ID of the network.
name	user-specified name of the network.
ipRange	IP range of the network.

role	role of the network, one of the following: • Supernet • Subnet • Range
zoneId	zone ID if applicable. Note: • a network can only belong to one zone. • multiple networks with the same IP range can exist within different zones. Consequently, the same IP addresses scanned with different zones will then result as different IP addresses, and subsequently can only be accessed from the corresponding zone /network. • ARP scans will inherit the zone from its routers.
description	user-specified description of the network.
dhcpServer	name of the DHCP server assigned to this network, if applicable.
scans	which scans are being used for this network, if any.
parentId	ID of the parent network, if applicable
usageMedium	% threshold value in which the usage will be yellow, by default a global value.
usageHigh	% threshold value in which the usage will be red, by default a global value.

skipNetwork	if the network is to be skipped for scanning ('don't scan network /address broadcast' in the UI), either 'true' or 'false'.
activeIps	number of scanned IPs that are active.
totalIps	total number of IPs covered by the network range.
conflict	names of other scopes that conflict with this one.
conflictTimestamp	timestamp of the conflict.
children	number of children of the network, if applicable.
fullPath	full path of IP range hierarchy to this network, useful for when networks have the same name but different parents, e.g. in the case of two networks both called 'test': parent 1/test parent 2/test
exceedsUsage	if the activeIps % exceeds the 'usageHigh' threshold.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/network/6?media=json</code>
--------------	--

OUTPUT	<pre>{ "id" : 1, "name" : "test", "ipRange" : "10.44.2.0/24", "role" : 0, "zoneID" : 0, "description" : "", "dhcpServer" : 0, "scans" : 3, "parentID" : 0, "usageMedium" : 0, "usageHigh" : 0, "skipNetwork" : true, "activeIps" : 124, "totalIps" : 256, "conflict" : null, "conflictTimestamp" : 0, "children" : [], "fullPath" : "test", "exceedsUsage" : false, }</pre>
---------------	--

PUT Method details

Modifies the settings of a specified network.

Request

See GET call table above.

Response

Updated network settings.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/network/6?media=json -X PUT -H "content-type:application/json" -d \ ' { "name" : "test", "ipRange" : "10.44.2.0/26", "role" : 0, "zoneID" : 0, "description" : "", "dhcpServer" : 0, "scans" : 3, "parentID" : 0, "usageMedium" : 0, "usageHigh" : 0, "skipNetwork" : true } '</pre>
OUTPUT	<pre>{ "id" : 1, "name" : "test", "ipRange" : "10.44.2.0/26", "role" : 0, "zoneID" : 0, "description" : "", "dhcpServer" : 0, "scans" : 3, "parentID" : 0, "usageMedium" : 0, "usageHigh" : 0, "skipNetwork" : true, "activeIps" : 124, "totalIps" : 256, "conflict" : null, "conflictTimestamp" : 0, "children" : [], "fullPath" : "test", "exceedsUsage" : false, }</pre>

DELETE Method details

Deletes the specified network.

Response

'OK' if successful.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/network/6?media=json -X DELETE</code>
OUTPUT	<pre>{ "message" : "OK" }</pre>

IPAM - IP Address Pages

List scanned IP addresses pages for a specified network.

URL: ipam/network/{id}/address

- [Method summary](#)
- [GET Method details](#)
 - [Example](#)

Method summary

- GET Method - list the scanned IP addresses page.

GET Method details

Returns a list of scanned IP addresses pages for a specified network. The index can be used as the page value for the 'IP Addresses' (ipam/network/{id}/address/{page} call. This is zero indexed, so the first entry has an index of 0, and should use /api/ipam/network/{id}/address/0 to get it.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/network/1/address?media=json</code>
OUTPUT	<code>["10.44.2.0 - 10.44.2.255"]</code>

IPAM - IP Addresses

List scanned IP addresses. To use a network's IP range and a page to get a list of IP addresses:

URL: ipam/network/{id}/address/{page}

To use a custom range to get a list of IP addresses (where range is "ipAddress-ipAddress"):

URL: ipam/address/{address}

URL: ipam/address/{address}?zoneId=#

- [Method Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - lists scanned IP addresses.

GET Method Details

Lists scanned IP addresses.

Response

Name	Description
zoneId	ID of the zone, if used.
comments	any comments added to this IP address by the user.
hostname	hostname, if applicable.
active	if the device has responded to a scan recently.
lastScan	last time the device was scanned.
lastActive	last time the device was successfully scanned.
history	number of history entries
ipv4	if the IP address is IPv4, either 'true' or 'false'.
ipAddress	IP address.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/network/1/address/0?media=json</code>
OUTPUT	<pre>[{ "zoneId" : 0, "comments" : null, "hostname" : null, "active" : true, "lastScan" : 1680520203, "lastActive" : 1680520203, "history" : null, "ipv4" : true, "ipAddress" : "10.44.2.33" }, { ... }, { "zoneID" : 0, "comments" : null, "hostname" : "test.entuity.local", "active" : true, "lastScan" : 1680520208, "lastActive" : 1680520208, "history" : null, "ipv4" : true, "ipAddress" : "10.44.2.195" }]</pre>

IPAM - IP Address Comments and History

List an IP address and its history, modify the comments on an IP address, and delete an IP address. To use a network's UP range, page and the address to get the address:

URL: ipam/network/{id}/address/{page}

To get the address directly:

URL: ipam/address/{address}

URL: ipam/address/{address}?zoneId=#

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Example](#)

Methods summary

- GET Method - returns an IP address and its history.
- PUT Method - modifies the comments for an IP address.
- DELETE Method - deletes an IP address.

GET Method details

Returns an IP address and its history.

Response

User info structure with the following format:

Name	Description
zoneId	zone of the DHCP server, if applicable. This column is hidden if there are no zones.
comments	any comments added to the IP address.
hostname	hostname, if applicable.

active	if the device has responded to a scan recently, either 'true' or 'false'.
lastScan	last time the device was scanned.
lastActive	last time the device was successfully scanned.
timestamp	timestamp of this IP history entry.
scan	scan type, one of the following: • 0 - Ping • 1 - Reverse DNS • 2 - ARP • 3 - DHCP
result	result of the scan, one of the following: • 0 - OK • 1 - Offline • 2 - Expiring
ipv4	if the IP address is IPv4, either 'true' or 'false'.
ipAddress	IP address.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/network/1/address/0/10.44.2.195?media=json</code>
--------------	--

OUTPUT	<pre>{ "zoneID" : 0, "comments" : null, "hostname" : "ent-17-0-lin.entuity.local", "active" : true, "lastScan" : 1680520208, "lastActive" : 1680520208, "history" : [{ "timestamp" : 1680520208, "scan" : 0, "result" : 0 }, { "timestamp" : 1680520208, "scan" : 1, "result" : 0 }], "ipv4" : true, "ipAddress" : "10.44.2.195" }</pre>
---------------	--

PUT Method details

Modifies comments for an IP address.

Request

Same as the GET call table above.

Response

Updated IP address result.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/network/1/address/0/10.44.2.195?media=json -X PUT -H "content-type:application/json" -d \ '{ "comments" : "test", }'</pre>
--------------	--

OUTPUT	<pre>{ "zoneID" : 0, "comments" : "test", "hostname" : "ent-17-0-lin.entuity.local", "active" : true, "lastScan" : 1680520208, "lastActive" : 1680520208, "history" : [{ "timestamp" : 1680520208, "scan" : 0, "result" : 0 }, { "timestamp" : 1680520208, "scan" : 1, "result" : 0 }], "ipv4" : true, "ipAddress" : "10.44.2.195" }</pre>
---------------	--

DELETE Method details

Deletes the specified IP address.

Response

'OK' if successful.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/network/1/address/0/10.44.2.195?media=json -X DELETE</pre>
OUTPUT	<pre>{ "message" : "OK" }</pre>

IPAM - DHCP Servers

Lists information about DHCP servers, and add a new DHCP server.

URL: ipam/dhcp

- [Methods Summary](#)
- [GET Method detail](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists DHCP servers.
- POST Method - creates a new DHCP server.

GET Method detail

Returns a list of DHCP servers.

Response

Response includes a list of DHCP servers. Each user has following attributes.

Name	Description
id	unique ID of the DHCP server.
name	user-specified name of the DHCP server.
ipAddress	IP address of the DHCP server.
zoneId	zone of the DHCP server, if applicable.
type	type of DHCP server, either 0 (Windows (WinRM)) or 1 (ISC (SSH)).
credId	ID of the shared credential set used for the DHCP server.
port	port used to connect to the DHCP server.
lastScanned	timestamp of the last scan on the DHCP server.

status	current status of the connection to the server, one of the following: • 0 - OK • 1 - No Response (e.g. offline or credentials invalid) • 2 - No DHCP (e.g. no DHCP installed)
(scope) name	name of the IP range.
ipRange	IP range of the scope.
active	if the IP range is active, either 'true' or 'false'.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/dhcp?media=json</code>
OUTPUT	<pre>[{ "id" : 1, "name" : "testServer", "ipAddress" : "10.44.2.49", "zoneId" : 0, "type" : 1, "credId" : 1, "port" : 0, "lastScanned" : 0, "status" : 0, "scopes" : null }]</pre>

POST Method details

Adds a new DHCP server.

Request

Name	Description
name	user-specified name of the DHCP server.
ipAddress	IP address of the DHCP server.
zoneId	zone of the DHCP server, if applicable.
type	type of DHCP server, either 0 (Windows (WinRM)) or 1 (ISC (SSH)).

credId	ID of the shared credential set used for the DHCP server.
port	port used to connect to the DHCP server.

Response

'OK' if successful.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/dhcp?media=json -X POST -H "content-type:application/json" -d \ { "name" : "testServer2", "ipAddress" : "10.44.2.37", "zoneId" : 0, "type" : 2, "credId" : 2, "port" : 0 }</pre>
OUTPUT	<pre>{ "message" : "OK" }</pre>

IPAM - DHCP Server Details

List information about a specified DHCP server, modify a DHCP server, and delete a DHCP server.

URL: ipam/dhcp/{id}

- [Methods summary](#)
- [GET Method details](#)
 - [Response](#)
 - [Example](#)
- [PUT Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method details](#)
 - [Response](#)
 - [Example](#)

Methods summary

- GET Method - returns information about the specified DHCP server.
- PUT Method - modifies DHCP server.
- DELETE Method - deletes DHCP server.

GET Method details

Returns information about the specified DHCP server.

Response

User info structure with the following format:

Name	Description
id	unique ID of the DHCP server.
name	user-specified name of the DHCP server.
ipAddress	IP address of the DHCP server.
zoneID	zone of the DHCP server, if applicable. This column is hidden if there are no zones.
type	type of DHCP server, either 0 (Windows (WinRM)) or 1 (ISC (SSH)).
credId	ID of the shared credential set used for the DHCP server.
port	port used to connect to the DHCP server.

lastScanned	timestamp of the last scan on the DHCP server.
status	current status of the connection to the server, one of the following: • 0 - OK • 1 - No Response (e.g. offline or credentials invalid) • 2 - No DHCP (e.g. no DHCP installed)
(scope) name	name of the IP range.
ipRange	IP range of the scope.
active	if the IP range is active, either 'true' or 'false'.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/ipam/dhcp/6?media=json</code>
OUTPUT	<pre>{ "id" : 1, "name" : "testServer", "ipAddress" : "10.44.2.49", "zoneID" : 0, "type" : 1, "credID" : 1, "port" : 0, "lastScanned" : 0, "status" : 0, "scopes" : [{ "name" : "testScope", "ipRange" : "10.44.1.0-10.44.1.255", "active" : true }] }</pre>

PUT Method details

Modifies details of the specified DHCP server.

Request

Name	Description
id	unique ID of the DHCP server.
name	user-specified name of the DHCP server.
ipAddress	IP address of the DHCP server.
zoneId	zone of the DHCP server, if applicable. This column is hidden if there are no zones.
type	type of DHCP server, either 0 (Windows (WinRM)) or 1 (ISC (SSH)).
credId	ID of the shared credential set used for the DHCP server.
port	port used to connect to the DHCP server.

Response

Detailed information about the DHCP server after changes, as the GET method would return it.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/dhcp/6?media=json -X PUT -H "content-type:application/json" -d \ '{ "id" : 1, "name" : "testServer2", "ipAddress" : "10.44.2.49", "zoneID" : 0, "type" : 1, "credID" : 1, "port" : 0 }'</pre>
-------	---

OUTPUT	<pre>{ "id" : 1, "name" : "testServer2", "ipAddress" : "10.44.2.49", "zoneID" : 0, "type" : 1, "credID" : 1, "port" : 0, "lastScanned" : 0, "status" : 0, "scopes" : [{ "name" : "testScope", "ipRange" : "10.44.1.0-10.44.1.255", "active" : true }] }</pre>
---------------	---

DELETE Method details

Deletes the specified DHCP server.

Response

'OK' if successful.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/ipam/dhcp/6?media=json -X DELETE</pre>
OUTPUT	<pre>{ "message" : "OK" }</pre>

Config Sets - List and Create Config Sets

List config sets, and create new config sets.

URL: `cfg/serverGroups`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists all config sets.
- POST Method - creates a new config set.

GET Method Details

Lists all config sets that have been created in a multi-server installation.

Response

Name	Description
syncUsers	if sync of users is enabled in this config set, either 'true' or 'false'.
syncUserGroups	if sync of user groups is enabled in this config set, either 'true' or 'false'.
syncViews	if sync of Views is enabled in this config set, either 'true' or 'false'.
serverGroupId	ID of the config set.
serverGroupName	name of the config set.
description	description of the config set, if any.
creationTimeSec	timestamp of when the config set was created.
createdBy	user who created the config set.
syncPaused	if synchronization of the config set is paused.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroups?media=json</code>
OUTPUT	<pre>{ "items" : [{ "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "serverGroupName" : "Kim1", "description" : "", "creationTimeSec" : 1676997135, "createdBy" : "kim1", "syncPaused" : false }, { "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true, "serverGroupId" : "8a938ad6-a2d8-4620-b3fc-ac8482f60731", "serverGroupName" : "Kim2", "description" : "", "creationTimeSec" : 1676998895, "createdBy" : "kim1", "syncPaused" : false }, { "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true, "serverGroupId" : "19b4d904-33be-4c54-a613-544c6ef691f8", "serverGroupName" : "Aardvark", "description" : "", "creationTimeSec" : 1677152823, "createdBy" : "kim1", "syncPaused" : false }, { "syncUsers" : false, "syncUserGroups" : false, "syncViews" : true,</pre>

```

    "serverGroupId" : "112810b8-265d-4c8d-930c-87f264581634",
    "serverGroupName" : "Test1",
    "description" : "",
    "creationTimeSec" : 1677240349,
    "createdBy" : "kim1",
    "syncPaused" : false
  } ],
  "count" : 5
}

```

POST Method details

Create a new config set.

Response

Name	Description
serverGroupName	name of the config set.
description	description of the config set, if any.
syncUsers	if sync of users is enabled in this config set, either 'true' or 'false'.
syncUserGroups	if sync of user groups is enabled in this config set, either 'true' or 'false'.
syncViews	if sync of Views is enabled in this config set, either 'true' or 'false'.

Example

INPUT	<pre> curl -u admin:admin https://localhost/api/cfg/serverGroups?media=json -X POST -H "content-type:application/json" -d \ { "serverGroupName": "First Group 1", "description": "Test group", "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true }' </pre>
--------------	--

OUTPUT	<pre>{ "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "serverGroupName" : "First Group 1", "description" : "Test group", "creationTimeSec" : 1677514241, "createdBy" : "admin", "syncPaused" : false }</pre>
---------------	--

Config Sets - Summary Info

List summary info about config sets.

URL: `cfg/serverGroupsSummaryInfo`

- [Method Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - lists all config sets and provides summary information about each.

GET Method Details

Lists all config sets and provides summary information about each.

Response

Name	Description
syncUsers	if sync of users is enabled in this config set, either 'true' or 'false'.
syncUserGroups	if sync of user groups is enabled in this config set, either 'true' or 'false'.
syncViews	if sync of Views is enabled in this config set, either 'true' or 'false'.
serverGroupId	ID of the config set.
serverGroupName	name of the config set.
description	description of the config set, if any.
creationTimeSec	timestamp of when the config set was created.
createdBy	user who created the config set.
syncPaused	if synchronization of the config set is paused.
configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
lastModifiedSec	timestamp of when the config set was last modified.

lastModifiedBy	user who last modified the config set.
memberCount	number of servers in the config set.
memberServerIds	IDs of the servers in the config set.
serverIdsAlreadySynced	IDs of the servers currently synced.
serverIdsQueuedToSync	IDs of the servers currently queued to sync.
failedServerIdsToErrorMsg	IDs of the servers that have failed to sync, and the explanatory error message for each server that has failed to sync.
syncStatusCategory	description of the config set's current sync status.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupsSummaryInfo</code>
OUTPUT	<pre>{ "items" : [{ "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "serverGroupName" : "Kim1", "description" : "", "creationTimeSec" : 1676997135, "createdBy" : "kim1", "syncPaused" : false, "configurationChangeId" : 76, "lastModifiedSec" : 1677511085, "lastModifiedBy" : "kim1", "memberCount" : 2, "syncStatus" : { "configurationChangeId" : 76, "memberServerIds" : ["d176aa00-d3d4-4ba1-9409-b480858f7a4d", "8acd818b-e0e7-4e8f-83d5-20070c942d3a"], "serverIdsAlreadySynced" : ["8acd818b-e0e7-4e8f-83d5-20070c942d3a"], "serverIdsQueuedToSync" : null, "failedServerIdsToErrorMsg" : { "d176aa00-d3d4-4ba1-9409-b480858f7a4d" : "Server ENTLONWBPC01 unreachable (last seen: 2023-02-27 15:15:23Z)" } } }] }</pre>

```
    },
    "syncPaused" : false
  },
  "syncStatusCategory" : "SYNC_PARTIAL_FAILURE"
}, {
  "syncUsers" : true,
  "syncUserGroups" : true,
  "syncViews" : true,
  "serverGroupId" : "8a938ad6-a2d8-4620-b3fc-ac8482f60731",
  "serverGroupName" : "Kim2",
  "description" : "",
  "creationTimeSec" : 1676998895,
  "createdBy" : "kim1",
  "syncPaused" : false,
  "configurationChangeId" : 55,
  "lastModifiedSec" : 1677229402,
  "lastModifiedBy" : "kim1",
  "memberCount" : 0,
  "syncStatus" : {
    "configurationChangeId" : 55,
    "memberServerIds" : null,
    "serverIdsAlreadySynced" : null,
    "serverIdsQueuedToSync" : null,
    "failedServerIdsToErrorMsg" : null,
    "syncPaused" : false
  },
  "syncStatusCategory" : "SYNC_NOTHING_TO_SYNCHRONIZE"
}, {
  "syncUsers" : true,
  "syncUserGroups" : true,
  "syncViews" : true,
  "serverGroupId" : "19b4d904-33be-4c54-a613-544c6ef691f8",
  "serverGroupName" : "Aardvark",
  "description" : "",
  "creationTimeSec" : 1677152823,
  "createdBy" : "kim1",
```

```
"syncPaused" : false,
"configurationChangeId" : 10,
"lastModifiedSec" : 1677152835,
"lastModifiedBy" : "kim1",
"memberCount" : 0,
"syncStatus" : {
  "configurationChangeId" : 10,
  "memberServerIds" : null,
  "serverIdsAlreadySynced" : null,
  "serverIdsQueuedToSync" : null,
  "failedServerIdsToErrorMsg" : null,
  "syncPaused" : false
},
"syncStatusCategory" : "SYNC_NOTHING_TO_SYNCHRONIZE"
}, {
  "syncUsers" : true,
  "syncUserGroups" : true,
  "syncViews" : true,
  "serverGroupId" : "5563b28c-ea24-4435-b6f3-0063377c0494",
  "serverGroupName" : "Funky",
  "description" : "",
  "creationTimeSec" : 1677153122,
  "createdBy" : "kim1",
  "syncPaused" : false,
  "configurationChangeId" : 12,
  "lastModifiedSec" : 1677153131,
  "lastModifiedBy" : "kim1",
  "memberCount" : 0,
  "syncStatus" : {
    "configurationChangeId" : 12,
    "memberServerIds" : null,
    "serverIdsAlreadySynced" : null,
    "serverIdsQueuedToSync" : null,
    "failedServerIdsToErrorMsg" : null,
    "syncPaused" : false
  },
```

```
"syncStatusCategory" : "SYNC_NOTHING_TO_SYNCHRONIZE"
}, {
  "syncUsers" : false,
  "syncUserGroups" : false,
  "syncViews" : true,
  "serverGroupId" : "112810b8-265d-4c8d-930c-87f264581634",
  "serverGroupName" : "Test1",
  "description" : "",
  "creationTimeSec" : 1677240349,
  "createdBy" : "kim1",
  "syncPaused" : false,
  "configurationChangeId" : 56,
  "lastModifiedSec" : 1677240349,
  "lastModifiedBy" : "kim1",
  "memberCount" : 0,
  "syncStatus" : {
    "configurationChangeId" : 56,
    "memberServerIds" : null,
    "serverIdsAlreadySynced" : null,
    "serverIdsQueuedToSync" : null,
    "failedServerIdsToErrorMsg" : null,
    "syncPaused" : false
  },
  "syncStatusCategory" : "SYNC_NOTHING_TO_SYNCHRONIZE"
} ],
"count" : 5
}
```

Config Sets - Update and Delete Config Sets

Update or delete a config set, and pause the synchronization of a config set.

URL: `cfg/serverGroups/{serverGroupId}`

- [Methods Summary](#)
- [PUT Method detail - update config set](#)
 - [Request](#)
 - [Example](#)
- [PUT Method detail - pause synchronization of a config set](#)
 - [Request](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- PUT Method - update a specified config set.
- PUT Method - pause synchronization for a specified config set
- DELETE Method - delete a specified config set.

PUT Method detail - update config set

Updates a specified config set.

Request

Name	Description
serverGroupName	name of the config set.
description	description of the config set, if any.
syncUsers	if sync of users is enabled in this config set, either 'true' or 'false'.
syncUserGroups	if sync of user groups is enabled in this config set, either 'true' or 'false'.
syncViews	if sync of Views is enabled in this config set, either 'true' or 'false'.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroups/82e7c418-cd48-4fbf-857b-cddb475841a2 -X PUT -H "content-type:application/json" -d '{ "serverGroupName": "First Group 1 changed", "description" : "My First Group changed", "syncUsers": true, "syncUserGroups" : true, "syncViews" : false }'</pre>
OUTPUT	<pre>{ "syncUsers" : true, "syncUserGroups" : true, "syncViews" : false, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "serverGroupName" : "First Group 1 changed", "description" : "My First Group changed", "creationTimeSec" : 1676997135, "createdBy" : "admin", "syncPaused" : false }</pre>

PUT Method detail - pause synchronization of a config set

Pause the synchronization of a specified config set. To trigger synchronization of a config set, please see **Server Configuration Sets - Sync Config Set**.

Request

Name	Description
syncPaused	if synchronization of the config set is paused, either 'true' or 'false'.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroups/82e7c418-cd48-4fbf-857b-cddb475841a2 -X PUT -H "content-type:application/json" -d { "syncPaused": true }</pre>
OUTPUT	<pre>{ "syncUsers" : true, "syncUserGroups" : true, "syncViews" : false, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "serverGroupName" : "First Group 1 changed", "description" : "My First Group changed", "creationTimeSec" : 1676997135, "createdBy" : "admin", "syncPaused" : true }</pre>

DELETE Method detail

Deletes a specified config set.

Response

'SUCCESS' when successfully deleted.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroups/6ed9ccd4-e0b8-4ee9-8f5d-c75974678c78?media=json -X DELETE</pre>
OUTPUT	<pre>{ "errorCode": "SUCCESS", "errorDetails": null, "data": { } }</pre>

Config Sets - Config Set Details

List and populate the configuration of a config set.

URL: `cfg/serverGroupsConfig/{serverGroupId}`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists all configuration details for a specified config set.
- POST Method - populate configuration details for a specified config set.

GET Method Details

Lists all config sets and provides summary information about each.

Response

Name	Description
syncUsers	if sync of users is enabled in this config set, either 'true' or 'false'.
syncUserGroups	if sync of user groups is enabled in this config set, either 'true' or 'false'.
syncViews	if sync of Views is enabled in this config set, either 'true' or 'false'.
serverGroupId	ID of the config set.
serverGroupName	name of the config set.
description	description of the config set, if any.
creationTimeSec	timestamp of when the config set was created.
createdBy	user who created the config set.
syncPaused	if synchronization of the config set is paused.
globalUserSettings	any global user settings that apply to this config set.

passwordComplexitySettings	any password complexity settings that apply to this config set.
username	username of a user in the config set.
password	the above user's password.
settings	individual user settings, e.g. timeout, account locking, password expiry.
passwordChangeld	ID of the change that created the current password.
userGroupName	user group assigned to the config set.
members	usernames assigned to this user group.
tools	string list of the tool permissions assigned to this user group, e.g "groupName": "Administrator Tools", "toolName": "Dashboard Management".
defaultReportPermission	default report permission assigned to this user group, one of the following: • NULL - no access • VIEW - view only • RUN - view and run • SCHEDULE - view, run and schedule • EDIT- view, run, schedule and edit
reportURLToPermission	cascading list of permissions for individual reports, e.g.: <pre>"reportURLToPermission": { "/reports/Virtualization/VirtualizationPerspective": "SCHEDULE" "/reports/Virtualization/vSwitchInventory": "RUN" }</pre>
reportFolderToPermission	cascading list of permissions for individual report folders, e.g.: <pre>"reportFolderToPermission": { "/reports/Activity": "RUN" "/reports/Administrative": "SCHEDULE" "/reports/Configuration": "RUN" }</pre>
defaultTaskPermission	default report permission assigned to this user group, one of the following: • NULL - no access • RUN - view and run • SCHEDULE - view, run and schedule
taskNameToPermission	cascading list of configuration management task permissions.
viewName	name of the View assigned to this config set.

baseViewType	if View content is based on union of the Views or intersection of the Views.
baseViewNames	names of the View that form the base of this current View.
contentsType	if View content is automatically (union/intersection) or manually determined.
domainFilter	name of the domain filter applied to this View.
eventFilter	name of the event filter applied to this View.
situationFilter	name of the incident filter applied to this View.
owner	user account that owns the user group. Can be 'system'.
groupName	name of the user group.
editable	if the user group is editable, either 'true' or 'false'.
longitude	longitude of the View, as specified for Geographical Map mode.
latitude	latitude of the View, as specified for Geographical Map mode.
locationName	name of the View's location, as specified for Geographical Map mode.
locationManuallySet	if the View's location is manually set, either 'true' or 'false'.
viewUUID	unique universal identifier of the View.
filterUUID	unique universal identifier of the filter.
name	name of the filter.
SRCTYPE	object source type to which this filter refers, one of the following: • 1024 - application • 4 - device • 1 - port • 65536 - service • 8 - VLAN
DEVTYPE	device type to which this filter refers.
PORTTYPE	port type to which this filter refers.
passIP	whether the filter should include devices not under management.
configurationChangeld	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
lastModifiedSec	timestamp of when the config set was last modified.
lastModifiedBy	user who last modified the config set.

Example

INPUT	curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2
OUTPUT	<pre>{ "syncUsers" : true, "syncUserGroups" : true, "syncViews" : true, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "serverGroupName" : "First Group 1", "description" : "Test group", "creationTimeSec" : 1677514241, "createdBy" : "admin", "syncPaused" : false, "configuration" : { "globalUserSettings" : null, "passwordComplexitySettings" : null, "users" : [{ "userName" : "admin", "password" : "2QFVD3sjU0l9IXHJskIaTyhg8fE=", "settings" : null, "passwordChangeId" : 77 }], { "userGroups" : [{ "userGroupName" : "Administrators", "members" : ["admin"], "tools" : null, "defaultReportPermission" : null, "reportURLToPermission" : null, "reportFolderToPermission" : null, "defaultTaskPermission" : null, "taskNameToPermission" : null }, { "userGroupName" : "All Users", "members" : ["admin", "user"], "tools" : null, "defaultReportPermission" : null, "reportURLToPermission" : null,</pre>

```

    "reportFolderToPermission" : null,
    "defaultTaskPermission" : null,
    "taskNameToPermission" : null
  } ],
  "views" : [ {
    "viewName" : "All Objects",
    "baseViewType" : 0,
    "baseViewNames" : [ ],
    "contentsType" : 2,
    "domainFilter" : "All Objects",
    "eventFilter" : "All Events",
    "situationFilter" : "All Incidents",
    "owner" : "<system>",
    "userGroups" : [ {
      "groupName" : "Administrators",
      "editable" : true
    } ],
    "longitude" : 0.0,
    "latitude" : 0.0,
    "locationName" : null,
    "locationManuallySet" : false,
    "viewUUID" : "--AllObjectsView--"
  }, {
    "viewName" : "Test1",
    "baseViewType" : 0,
    "baseViewNames" : [ ],
    "contentsType" : 1,
    "domainFilter" : "All Objects",
    "eventFilter" : "All Events",
    "situationFilter" : "All Incidents",
    "owner" : "<system>",
    "userGroups" : [ {
      "groupName" : "Administrators",
      "editable" : true
    } ],
    {
      "groupName" : "All Users",

```

```
        "editable" : false
    } ],
    "longitude" : 0.0,
    "latitude" : 0.0,
    "locationName" : null,
    "locationManuallySet" : false,
    "viewUUID" : "63007757-8a49-4258-b647-24802a8a4bf3"
} ],
"contentFilters" : [ {
    "filterUUID" : "--AllObjectsDomain--",
    "name" : "All Objects"
}, {
    "filterUUID" : "--InfrastructureOnlyDomain--",
    "name" : "Infrastructure Only",
    "rules" : [ {
        "SRCTYPE" : "4"
    }, {
        "SRCTYPE" : "1024"
    }, {
        "SRCTYPE" : "8"
    }, {
        "SRCTYPE" : "1",
        "DEVTYPE" : "168"
    }, {
        "SRCTYPE" : "1",
        "DEVTYPE" : "148",
        "PORTTYPE" : "2"
    }, {
        "SRCTYPE" : "1",
        "DEVTYPE" : "148",
        "PORTTYPE" : "3"
    } ]
} ],
"eventFilters" : [ {
    "filterUUID" : "--AllEventsFilter--",
    "name" : "All Events",
```

```

    "passIP" : true
  } ],
  "incidentFilters" : [ {
    "filterUUID" : "--AllIncidentsFilter--",
    "name" : "All Incidents",
    "passIP" : true
  } ]
},
"configurationChangeId" : 78,
"lastModifiedSec" : 1677514577,
"lastModifiedBy" : "kim1"
}

```

POST Method Details

Populate configuration details for a specified config set.

Request

See the 'Response' table for GET above.

Response

On success, displays the config definition.

Example

INPUT	<pre> curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418- cd48-4fbf-857b-cddb475841a2 -X POST -H "content-type:application/json" -d { "globalUserSettings" : null, "passwordComplexitySettings" : null, "users" : [{ "userName" : "admin", "password" : "2QFVD3sjU0l9IXHJskIaTyhg8fE=", "settings" : null, "passwordChangeId" : 91 }, { "userName" : "user", </pre>
-------	--

```
"password" : "jUx3/xeXPVXQ132Q1gJdbh7yc9Q=",
"settings" : null,
"passwordChangeId" : 91
} ],
"userGroups" : [ {
  "userGroupName" : "Administrators",
  "members" : [ "admin" ],
  "tools" : null,
  "defaultReportPermission" : null,
  "reportURLToPermission" : null,
  "reportFolderToPermission" : null,
  "defaultTaskPermission" : null,
  "taskNameToPermission" : null
}, {
  "userGroupName" : "All Users",
  "members" : [ "admin", "user" ],
  "tools" : null,
  "defaultReportPermission" : null,
  "reportURLToPermission" : null,
  "reportFolderToPermission" : null,
  "defaultTaskPermission" : null,
  "taskNameToPermission" : null
} ],
"views" : [ {
  "viewName" : "All Objects",
  "baseViewType" : 0,
  "baseViewNames" : [ ],
  "contentsType" : 2,
  "domainFilter" : "All Objects",
  "eventFilter" : "All Events",
  "situationFilter" : "All Incidents",
  "owner" : "<system>",
  "userGroups" : [ {
    "groupName" : "Administrators",
    "editable" : true
  } ],
}
```

```
"longitude" : 0.0,
"latitude" : 0.0,
"locationName" : null,
"locationManuallySet" : false,
"viewUUID" : "--AllObjectsView--"
} ],
"contentFilters" : [ {
  "filterUUID" : "--AllObjectsDomain--",
  "name" : "All Objects"
}, {
  "filterUUID" : "--InfrastructureOnlyDomain--",
  "name" : "Infrastructure Only",
  "rules" : [ {
    "SRCTYPE" : "4"
  }, {
    "SRCTYPE" : "1024"
  }, {
    "SRCTYPE" : "8"
  }, {
    "SRCTYPE" : "1",
    "DEVTYPE" : "168"
  }, {
    "SRCTYPE" : "1",
    "DEVTYPE" : "148",
    "PORTTYPE" : "2"
  }, {
    "SRCTYPE" : "1",
    "DEVTYPE" : "148",
    "PORTTYPE" : "3"
  } ]
} ],
"eventFilters" : [ {
  "filterUUID" : "--AllEventsFilter--",
  "name" : "All Events",
  "passIP" : true
} ],
```

```
"incidentFilters" : [ {  
  "filterUUID" : "--AllIncidentsFilter--",  
  "name" : "All Incidents",  
  "passIP" : true  
} ]  
'
```

OUTPUT [on success, displays the config definition]

Config Sets - Config Set Active Config

List active config from a server, determined by the optional parameter(s)

URL: `cfg/serverConfig?{parameter=true}&{parameter=true}&{parameter=true}`

- [Method Summary](#)
- [GET Method Details](#)
 - [Request](#)
 - [Example](#)

Method Summary

- GET Method - lists active config from a server, determined by the selected parameters.

GET Method Details

Lists active config from a server, determined by the selected parameters.

Request

Specify the parameters used to populate the list, choosing from:

- `includeUsers`
- `includeUserGroups`
- `includeViews`
- `includePasswords`

If passwords are included, the output only returns the encoded form of the passwords.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverConfig?includeUsers=true&includeUserGroups=true&includeViews=true&includePasswords=true&media=json</code>
OUTPUT	[on success, displays the config definition]

Config Sets - Populate Config Set from Local Server

Populate the configuration of a config set from a local server. This is two RESTful API calls combined.

URL: `cfg/serverConfig?{parameter=true}&{parameter=true}&{parameter=true} | cfg/serverConfig/{serverGroupId}`

- [Methods Summary](#)
- [GET and POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists the parameters from the local server with which to populate a config set; and POST Method - add these parameters to a specified config set.

GET and POST Method Details

Request

In the GET call, specify parameters e.g. `includeUsers=true`, `includeViews=true`, that you wish to then POST to the specified config set.

Response

When GET call is made, the output is the config definition. This can then be added to the call to specify the config on a config set.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverConfig?includeUsers=true&includeUserGroups=true&includeViews=true&includePasswords=true curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-c4fbf-857b-cddb475841a2/ -X POST -H "content-type:application/json" -d @-</pre>
OUTPUT	[config set definition]

Config Sets - Servers

List servers currently assigned to a specified config set, and add and remove servers from a config set.

URL: `cfg/serverGroupMembership/{serverGroupId}`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method details - add a server to a config set](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [POST Method details - remove a server from a config set](#)
 - [Request](#)
 - [Example](#)

Methods Summary

- GET Method - lists servers currently assigned to a specified config set.
- POST Method - add a server to a config set.
- POST Method - remove a server from a config set.

GET Method Details

Lists servers currently assigned to a specified config set.

Response

Name	Description
serverId	unique server identifier.
serverGroupId	config set ID.
count	number of servers assigned to the config set.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupMembership/bb816bfa-4f95-4337-9618-de220e055905?media=json</code>
-------	--

OUTPUT	<pre>{ "items" : [{ "serverId" : "8acd818b-e0e7-4e8f-83d5-20070c942d3a", "serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905", }, { "count" : 1 } }</pre>
---------------	--

POST Method details - add a server to a config set

Add a server to a config set.

Request

Specify the server(s) to be added using the 'membersAdded' parameter, listing the serverId.

Response

Success or error. An error may arise if the server that is being added to the config set already belongs to another config set. Please see 'Config Sets - Force Control of Server' to force control of servers in this case.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupMembership /82e7c418-cd48-4fbf-857b-cddb475841a2?media=json -X POST -H "content-type: application/json" -d \ { "membersAdded": ["8acd818b-e0e7-4e8f-83d5-20070c942d3a"] }'</pre>
OUTPUT	<pre>{"code":412,"reasonPhrase":"Precondition Failed","description":"Error modifying Entuity Server Group membership: serverId=8acd818b-e0e7-4e8f- 83d5-20070c942d3a, errorCode=ERROR_OPERATION_FAILURE. Server is already controlled: controllingServer=SOARQUIST, controllingGroup=Kim1"}</pre>

POST Method details - remove a server from a config set

Remove a server from a config set.

Request

Specify the server(s) to be removed using the 'membersRemoved' parameter, listing the serverId.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupMembership /82e7c418-cd48-4fbf-857b-cddb475841a2?media=json -X POST -H "content-type: application/json" -d \ { "membersRemoved": ["d176aa00-d3d4-4ba1-9409-b480858f7a4d"] }'</pre>
OUTPUT	<pre>{ "items" : [], "count" : 0 }</pre>

Config Sets - Force Control of Server

Force control of a server (that already belongs to a different config set) when adding it to a new config set.

URL: `cfg/serverGroupMembership/{serverId}?forceControl=true`

- [Method Summary](#)
- [POST Method details](#)
 - [Example](#)

Method Summary

- POST Method - force control of a server (that already belongs to a different config set) when adding it to a new config set.

POST Method details

Force control of a server (that already belongs to a different config set) when adding it to a new config set.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupMembership /bb816bfa-4f95-4337-9618-de220e055905?forceControl=true?media=json -X POST -H "content-type:application/json" -d \ { "membersAdded": ["d176aa00-d3d4-4ba1-9409-b480858f7a4d"] }'</pre>
OUTPUT	<pre>{ "items" : [{ "serverId" : "d176aa00-d3d4-4ba1-9409-b480858f7a4d", "serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905" }], "count" : 1 }</pre>

Config Sets - Sync

Trigger synchronization for a specified config set.

URL: `cfg/serverGroupSync/{serverGroupId}`

- [Method Summary](#)
- [POST Method Details](#)
 - [Example](#)

Method Summary

- POST Method - trigger the synchronization of a config set.

POST Method Details

Triggers the synchronization of a config set. To pause synchronization of a config set, please see **Server Configuration Sets - Update and Delete Config Sets**.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupSync/82e7c418-cd48-4fbf-857b-cddb475841a2 -X POST -H "content-type: application/json" -d \</pre>
-------	--

Config Sets - Sync (Change) History

List sync (change) history for all config sets, or filter by individual config set and/or member servers.

URL: `cfg/serverGroupConfigChanges?{serverGroupId}or{memberServerId}`

- [Method Summary](#)
- [GET Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Method Summary

- GET Method - lists sync (change) history of all config sets, or filter by individual config sets and/or member servers.

GET Method Details

Lists changes to a config set, filtered by the config set itself or a member server.

Request

The optional parameters are:

- `serverGroupId`
- `memberServerId`

Response

Name	Description
serverId	unique ID of the server.
configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
timestampSec	timestamp of the change
triggeredBy	user who made the change.

errorCode	code for the change outcome, e.g. 'Success'.
errorMsg	details of an error, if applicable.
serverGroupID	ID of the config set.
serverGroupName	name of the config set.
usersTotal	total number of users after this change on this config set or server.
usersRemoved	number of users removed by this change from this config set or server.
usersAdded	number of users added by this change to this config set or server.
userGroupsTotal	number of user groups after this change on this config set or server.
userGroupsRemoved	number of user groups removed by this change from this config set or server.
userGroupsAdded	number of user groups added by this change to this config set or server.
viewsTotal	number of Views after this change on this config set or server.
viewsRemoved	number of Views removed by this change from this config set or server.
viewsAdded	number of Views added by this change to this config set or server.
count	total number of results from this call.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroups?media=json</code>
OUTPUT	<pre>{ "items" : [{ "serverId" : "8acd818b-e0e7-4e8f-83d5-20070c942d3a", "configurationChangeId" : 73, "timestampSec" : 1677510898, "triggeredBy" : "kim1",</pre>

```
"errorCode" : "SUCCESS",
"errorMsg" : null,
"serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
"serverGroupName" : "Kim1",
"usersTotal" : 8,
"usersRemoved" : 1,
"usersAdded" : 0,
"userGroupsTotal" : 10,
"userGroupsRemoved" : 0,
"userGroupsAdded" : 0,
"viewsTotal" : 30,
"viewsRemoved" : 0,
"viewsAdded" : 0
}, {
  "serverId" : "8acd818b-e0e7-4e8f-83d5-20070c942d3a",
  "configurationChangeId" : 76,
  "timestampSec" : 1677511085,
  "triggeredBy" : "kim1",
  "errorCode" : "SUCCESS",
  "errorMsg" : null,
  "serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
  "serverGroupName" : "Kim1",
  "usersTotal" : 9,
  "usersRemoved" : 0,
  "usersAdded" : 1,
  "userGroupsTotal" : 10,
  "userGroupsRemoved" : 0,
  "userGroupsAdded" : 0,
  "viewsTotal" : 30,
  "viewsRemoved" : 0,
  "viewsAdded" : 0
}, {
  "serverId" : "d176aa00-d3d4-4ba1-9409-b480858f7a4d",
  "configurationChangeId" : 73,
  "timestampSec" : 1677510898,
  "triggeredBy" : "kim1",
```

```
"errorCode" : "SUCCESS",
"errorMsg" : null,
"serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
"serverGroupName" : "Kim1",
"usersTotal" : 8,
"usersRemoved" : 2,
"usersAdded" : 0,
"userGroupsTotal" : 10,
"userGroupsRemoved" : 0,
"userGroupsAdded" : 0,
"viewsTotal" : 30,
"viewsRemoved" : 0,
"viewsAdded" : 0
}, {
  "serverId" : "d176aa00-d3d4-4ba1-9409-b480858f7a4d",
  "configurationChangeId" : 74,
  "timestampSec" : 1677510940,
  "triggeredBy" : "kim1",
  "errorCode" : "SUCCESS",
  "errorMsg" : null,
  "serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
  "serverGroupName" : "Kim1",
  "usersTotal" : 9,
  "usersRemoved" : 0,
  "usersAdded" : 1,
  "userGroupsTotal" : 10,
  "userGroupsRemoved" : 0,
  "userGroupsAdded" : 0,
  "viewsTotal" : 30,
  "viewsRemoved" : 0,
  "viewsAdded" : 0
}, {
  "serverId" : "8acd818b-e0e7-4e8f-83d5-20070c942d3a",
  "configurationChangeId" : 74,
  "timestampSec" : 1677510940,
  "triggeredBy" : "kim1",
```

```
"errorCode" : "SUCCESS",
"errorMsg" : null,
"serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
"serverGroupName" : "Kim1",
"usersTotal" : 9,
"usersRemoved" : 0,
"usersAdded" : 1,
"userGroupsTotal" : 10,
"userGroupsRemoved" : 0,
"userGroupsAdded" : 0,
"viewsTotal" : 30,
"viewsRemoved" : 0,
"viewsAdded" : 0
}, {
  "serverId" : "d176aa00-d3d4-4ba1-9409-b480858f7a4d",
  "configurationChangeId" : 75,
  "timestampSec" : 1677510992,
  "triggeredBy" : "kim1",
  "errorCode" : "ERROR_SYNCHRONIZATION_FAILURE",
  "errorMsg" : "Server ENTLONWBPC01 unreachable (last seen: 2023-02-27
15:15:23Z)",
  "serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
  "serverGroupName" : "Kim1",
  "usersTotal" : null,
  "usersRemoved" : null,
  "usersAdded" : null,
  "userGroupsTotal" : null,
  "userGroupsRemoved" : null,
  "userGroupsAdded" : null,
  "viewsTotal" : null,
  "viewsRemoved" : null,
  "viewsAdded" : null
}, {
  "serverId" : "8acd818b-e0e7-4e8f-83d5-20070c942d3a",
  "configurationChangeId" : 75,
  "timestampSec" : 1677510992,
  "triggeredBy" : "kim1",
```

```
"errorCode" : "SUCCESS",
"errorMsg" : null,
"serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
"serverGroupName" : "Kim1",
"usersTotal" : 8,
"usersRemoved" : 1,
"usersAdded" : 0,
"userGroupsTotal" : 10,
"userGroupsRemoved" : 0,
"userGroupsAdded" : 0,
"viewsTotal" : 30,
"viewsRemoved" : 0,
"viewsAdded" : 0
}, {
  "serverId" : "d176aa00-d3d4-4ba1-9409-b480858f7a4d",
  "configurationChangeId" : 76,
  "timestampSec" : 1677511085,
  "triggeredBy" : "kim1",
  "errorCode" : "ERROR_SYNCHRONIZATION_FAILURE",
  "errorMsg" : "Server ENTLONWBPC01 unreachable (last seen: 2023-02-27
15:15:23Z)",
  "serverGroupId" : "bb816bfa-4f95-4337-9618-de220e055905",
  "serverGroupName" : "Kim1",
  "usersTotal" : null,
  "usersRemoved" : null,
  "usersAdded" : null,
  "userGroupsTotal" : null,
  "userGroupsRemoved" : null,
  "userGroupsAdded" : null,
  "viewsTotal" : null,
  "viewsRemoved" : null,
  "viewsAdded" : null
} ],
"count" : 8
}
```

Config Sets - Users

List all users in a config set, and add users to a config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/users`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Example](#)

Methods Summary

- GET Method - lists all users in a config set.
- POST Method - adds a user to a config set.

GET Method Details

Lists all users in a specified config set.

Response

Name	Description
username	username.
password	user's password.
settings	individual user settings, e.g. timeout, account locking, password expiry.
passwordChangeId	ID of the change that created the current password.
count	number of users returned in this config set.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/users?media=json</code>
-------	--

```

OUTPUT {
  "items" : [ {
    "userName" : "admin",
    "password" : "2QFVD3sjU0l9IXHJskIaTyhg8fE=",
    "settings" : null,
    "passwordChangeId" : 77
  }, {
    "userName" : "user",
    "password" : "jUx3/xeXPVXQ132Q1gJdbh7yc9Q=",
    "settings" : null,
    "passwordChangeId" : 77
  } ],
  "count" : 2
}

```

POST Method Details

Add a user to a specified config set. Note, a user can be added without necessarily specifying settings (i.e., a user can be added using only 'userName' and 'password').

Request

Name	Description
username	username.
password	user's password.
timeoutMinutes	number of minutes of inactivity after which the user will be logged out.
lockAttempts	number of failed attempts to log in after which the user will be locked out, -1 to disable.
lockDurationFailedAttempts	number of minutes for which the user will be locked out after exceeding the lockAttempts figure. -1 disables this, and it will revert to global settings, if any.
expiryDays	number of days after which the user account will expire. -1 disables this.
pwChangeDays	number of days until password change. -1 disables this.
passwordExpiryNoticePeriod	how many days' notice during which a password expiration warning will be displayed before the password expires.
overridePasswordSettings	if the global user password settings are overridden, either 'true' or 'false'.

forcePWChange	specify whether to force the user to change their password the next time they log in, either 'true' or 'false'.
errorCode	code for the change outcome, e.g. 'Success'.
errorDetails	details of an error, if applicable.
configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
serverGroupId	ID of the config set to which the user was added.
timestampSec	timestamp of when this config set was modified.
modifiedBy	user who made this change.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418- cd48-4fbf-857b-cddb475841a2/users -X POST -H "content-type:application /json" -d ' { "username" : "guy11", "password" : "supersecret" "settings" : { "timeoutMinutes" : 60, "lockAttempts" : 3, "lockDurationFailedAttempts" : -1, "expiryDays" : 2, "pwChangeDays" : 30, "passwordExpiryNoticePeriod" : 0, "overridePasswordSettings" : true, "forcePWChange" : false } }'</pre>
--------------	--

OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 79, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1677514827, "modifiedBy" : "admin" } }</pre>
---------------	---

Config Sets - User Details

Change a user's password, or delete a user from a config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/users/{username}`

- [Methods Summary](#)
- [PUT Method detail](#)
 - [Parameters](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- PUT Method - change a user's password.
- DELETE Method - delete a user from a config set.

PUT Method detail

Change a user's password.

Parameters

Name	Description
password	new password.
username	specified user.
settings	individual user settings, e.g. timeout, account locking, password expiry.
passwordChangeId	ID of the change that created the current password.

Response

In all cases, the new password will be encrypted in the output.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/bb816bfa-4f95-4337-9618-de220e055905/users/guy9 -X PUT -H "content-type:application/json" -d '{ "password": "evenmoresecret" }'</pre>
OUTPUT	<pre>{ "userName" : "guy9", "password" : "7mX01i0okUVFUZGNFUN109f0ygk=", "settings" : null, "passwordChangeId" : 128, }</pre>

DELETE Method detail

Deletes a user from a config set.

Response

'SUCCESS' when successfully deleted.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/users/guy9?media=json -X DELETE</pre>
OUTPUT	<pre>{ "errorCode": "SUCCESS", "errorDetails": "user 'guy9' removed" }</pre>

Config Sets - User Group Membership

Modifies user membership of a specified user group on a specified config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/userGroups/{userGroupName}`

Add users to a specified user group on a specified config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/userGroups/{userGroupName}/members`

Delete a user from a specified user group on a specified config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/userGroups/{userGroupName}/members/{username}`

- [Method Summary](#)
- [PUT Method details](#)
 - [Parameters](#)
 - [Example](#)
- [POST Method details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Response](#)
 - [Example](#)

Method Summary

- PUT Method - modifies the user membership of a user group on a config set.
- POST Method - add users to a specified user group on a specified config set.
- DELETE Method - delete a user from a specified user group on a specified config set.

PUT Method details

Modifies the user membership of a user group on a config set.

Parameters

Name	Description
userGroupName	name of the user group for which the membership is to be modified.
members	list the users who will now become members of this user group. This list overwrites the previous users in the user group.

errorCode	code for the change outcome, e.g. 'Success'.
errorDetails	details of an error, if applicable.
configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
serverGroupId	ID of the config set.
timestampSec	timestamp of the change.
modifiedBy	user who made the change.

Example

INPUT	<pre>curl -u admin:admin -k https://localhost/api/cfg/serverGroupConfig /82e7c418-cd48-4fbf-857b-cddb475841a2/userGroups/Administrators -X PUT -H 'content-type:application/json' { "userGroupName" : "24", "members" : ["kim1", "admin", "super1"] }</pre>
OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 100, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1677587075, "modifiedBy" : "admin" } }</pre>

POST Method details

Add users to a specified user group on a specified config set.

Request

Specify the username and password of the user to be added.

Response

'SUCCESS' when successfully added.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/userGroups/usergroup1/members?media=json -X POST -H "content-type:application/json" -d \ { "username" : "guy9" "password" : "supersecret" }'</pre>
OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 79, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1677514827, "modifiedBy" : "admin" } }</pre>

DELETE Method detail

Deletes a user from a specified user group on a specified config set

Response

'SUCCESS' when successfully deleted.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/userGroups/userGroup1/members/guy9?media=json -X DELETE</pre>
--------------	---

OUTPUT	<pre>{ "errorCode": "SUCCESS", "errorDetails": "user 'guy9' removed" }</pre>
---------------	--

Config Sets - List and Add Views

List the Views in a config set, and add Views to a config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/views`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Methods Summary

- GET Method - lists the Views in a specified config set.
- POST Method - add a new top-level View or Subview to a specified config set.

GET Method Details

Lists the Views in a specified config set.

Response

Name	Description
viewName	name of the View assigned to this config set.
baseViewType	if View content is based on union of the Views or intersection of the Views.
baseViewNames	names of the View that form the base of this current View.
contentsType	if View content is automatically (union/intersection) or manually determined.
domainFilter	name of the domain filter applied to this View.
eventFilter	name of the event filter applied to this View.
situationFilter	name of the incident filter applied to this View.
owner	user account that owns the user group. Can be 'system'.
groupName	name of the user group.

editable	if the user group is editable, either 'true' or 'false'.
longitude	longitude of the View, as specified for Geographical Map mode.
latitude	latitude of the View, as specified for Geographical Map mode.
locationName	name of the View's location, as specified for Geographical Map mode.
locationManuallySet	if the View's location is manually set, either 'true' or 'false'.
viewUUID	unique universal identifier of the View.
count	number of Views in the config set.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/views?media=json</code>
OUTPUT	<pre>{ "items" : [{ "items" : [{ "viewName" : "All Objects", "baseViewType" : 0, "baseViewNames" : [], "contentsType" : 2, "domainFilter" : "All Objects", "eventFilter" : "All Events", "situationFilter" : "All Incidents", "owner" : "<system>", "userGroups" : [{ "groupName" : "Administrators", "editable" : true }], "longitude" : 0.0, "latitude" : 0.0, "locationName" : null, "locationManuallySet" : false, "viewUUID" : "--AllObjectsView--" }], { "viewName" : "Test1", "baseViewType" : 0, "baseViewNames" : [],</pre>

```
"contentType" : 1,
"domainFilter" : "All Objects",
"eventFilter" : "All Events",
"situationFilter" : "All Incidents",
"owner" : "<system>",
"userGroups" : [ {
  "groupName" : "Administrators",
  "editable" : true
}, {
  "groupName" : "All Users",
  "editable" : false
} ],
"longitude" : 0.0,
"latitude" : 0.0,
"locationName" : null,
"locationManuallySet" : false,
"viewUUID" : "63007757-8a49-4258-b647-24802a8a4bf3"
} ],
"count" : 2
}
```

POST Method Details

Add a new top-level View or Subview to a specified config set

Request

See the 'Response' table for GET above.

Response

Displays success or error.

Examples

To add a top-level View:

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/views -X POST -H "content-type:application/json" -d '{ "viewName" : "All The Stuff", "baseViewType" : 1, "baseViewNames" : ["All Objects"], "contentsType" : 2, "domainFilter" : "All Objects", "eventFilter" : "Test Event Filter 1", "situationFilter" : "All Incidents", "owner" : "kim1", "userGroups" : [{ "groupName" : "Administrators", "editable" : true }], "longitude" : 0.0, "latitude" : 0.0, "locationName" : "", "locationManuallySet" : false }'</pre>
OUTPUT	<pre>{"code":412,"reasonPhrase":"Precondition Failed","description":"Error adding view to Entuity Server Group configuration: No event filter named 'Test Event Filter 1' exists"}</pre>

To add a Subview:

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/views -X POST -H "content-type:application/json" -d '{ "viewName" : "All Customers/Customer Zero", "baseViewType" : 0, "baseViewNames" : [], "contentsType" : 1, "domainFilter" : "All Objects", "eventFilter" : "Another Filter", "situationFilter" : "All Incidents", "owner" : "kim1", "userGroups" : [{ "groupName" : "Administrators", "editable" : true }], "longitude" : 0.0, "latitude" : 0.0, "locationName" : "", "locationManuallySet" : false }'</pre>
OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 197, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1680198645, "modifiedBy" : "admin" } }</pre>

Config Sets - Update, Rename and Delete Views

Update, rename or delete Views in a config set.

URL: `cfg/serverGroupConfig/{serverGroupId}/views/{viewUUID}`

- [Method Summary](#)
- [PUT Method detail - update View](#)
 - [Request](#)
 - [Example](#)
- [PUT Method detail - rename View](#)
 - [Response](#)
 - [Example](#)
- [DELETE Method detail](#)
 - [Request](#)
 - [Response](#)
 - [Examples](#)

Method Summary

- PUT Method - updates a View.
- PUT Method - renames a View.
- DELETE Method - delete a View from a specified config set.

PUT Method detail - update View

Updates a specified View.

Request

Name	Description
viewName	name of the View.
eventFilter	name of the event filter applied to this View.
situationFilter	name of the incident filter applied to this View.
owner	user account that owns the user group. Can be 'system'.
groupName	name of the user group.
editable	if the user group is editable, either 'true' or 'false'.
longitude	longitude of the View, as specified for Geographical Map mode.

latitude	latitude of the View, as specified for Geographical Map mode.
locationName	name of the View's location, as specified for Geographical Map mode.
locationManuallySet	if the View's location is manually set, either 'true' or 'false'.
viewUUID	number of Views in the config set.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418- cd48-4fbf-857b-cddb475841a2/views/95ff210b-6249-4573-b38e-c957cffad4ad -X PUT -H "content-type:application/json" -d ' { "viewName" : "T2", "eventFilter" : "Test Event Filter", "situationFilter" : "Test Incident Filter", "owner" : "admin", "userGroups" : [{ "groupName" : "Administrators", "editable" : true }, { "groupName" : "Guys", "editable" : false }] }</pre>
--------------	--

```

OUTPUT {
  "viewName" : "T2",
  "baseViewType" : 0,
  "baseViewNames" : [ ],
  "contentsType" : 1,
  "domainFilter" : "All Objects",
  "eventFilter" : "Test Event Filter",
  "situationFilter" : "Test Incident Filter",
  "owner" : "admin",
  "userGroups" : [ {
    "groupName" : "Administrators",
    "editable" : true
  }, {
    "groupName" : "Guys",
    "editable" : false
  } ],
  "longitude" : 0.0,
  "latitude" : 0.0,
  "locationName" : "",
  "locationManuallySet" : false,
  "viewUUID" : "95ff210b-6249-4573-b38e-c957cffad4ad"
}

```

PUT Method detail - rename View

Renames a specified View.

Response

See the table for PUT Method above.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/views/95ff210b-6249-4573-b38e-c957cffad4ad -X PUT -H "content-type:application/json" -d '{ "viewName" : "T2 renamed", }'</pre>
OUTPUT	<pre>{ "viewName" : "T2", "baseViewType" : 0, "baseViewNames" : [], "contentsType" : 1, "domainFilter" : "All Objects", "eventFilter" : "Test Event Filter", "situationFilter" : "Test Incident Filter", "owner" : "admin", "userGroups" : [{ "groupName" : "Administrators", "editable" : true }, { "groupName" : "Guys", "editable" : false }], "longitude" : 0.0, "latitude" : 0.0, "locationName" : "", "locationManuallySet" : false, "viewUUID" : "95ff210b-6249-4573-b38e-c957cffad4ad" }</pre>

DELETE Method detail

Deletes a View from a specified config set.

Request

Specify a View by its UUID or View name.

Response

'SUCCESS' when successfully deleted.

Examples

Specifying a View by its UUID:

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/views/38aeb745-000c-411f-a987-c761622aab85?media=json -X DELETE</code>
OUTPUT	<pre>{ "errorCode": "SUCCESS", "errorDetails": "view 'Test1' removed" }</pre>

Specifying a View by its View name:

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/views/Test1?media=json -X DELETE</code>
OUTPUT	<pre>{ "errorCode": "SUCCESS", "errorDetails": "view 'Test1' removed" }</pre>

Config Sets - Content Filters

List the content filters in a config set, and add or update content filters.

URL: `cfg/serverGroupsConfig/{serverGroupId}/contentFilters`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists the content filters applied to a specified config set.
- POST Method - add or update content filters in a specified config set.

GET Method Details

Lists the content filters applied to a specified config set.

Response

Name	Description
filterUUID	unique filter identifier.
name	name of the filter.
SRCTYPE	object source type to which this filter refers, one of the following: • 1024 - application • 4 - device • 1 - port • 65536 - service • 8 - VLAN
DEVTYPE	device type to which this filter refers.
PORTTYPE	port type to which this filter refers.
count	number of content filters listed.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/contentFilters</pre>
OUTPUT	<pre>{ "items" : [{ "filterUUID" : "--AllObjectsDomain--", "name" : "All Objects" }, { "filterUUID" : "--InfrastructureOnlyDomain--", "name" : "Infrastructure Only", "rules" : [{ "SRCTYPE" : "4" }, { "SRCTYPE" : "1024" }, { "SRCTYPE" : "8" }, { "SRCTYPE" : "1", "DEVTYPE" : "168" }, { "SRCTYPE" : "1", "DEVTYPE" : "148", "PORTTYPE" : "2" }, { "SRCTYPE" : "1", "DEVTYPE" : "148", "PORTTYPE" : "3" }] }, { "filterUUID" : "81797c36-49d1-41c5-8f4d-4fc82ef6b301", "name" : "Test Content Filter" }, { "filterUUID" : "549fb464-7364-4569-b1d5-8d32929563ae", "name" : "Test Content Filter1 - edit4" }, { "filterUUID" : "25cb3538-b918-43f6-adc9-d56b9cd3f20b",</pre>

```

    "name" : "Test Content Filter2 - renamed properly"
  }, {
    "filterUUID" : "62772711-c215-452e-bda6-208f86ab6db4",
    "name" : "Test Content Filter1 - edited"
  }, {
    "filterUUID" : "546d8f14-1c0b-4277-bc02-0b36ee0b9e2d",
    "name" : "Test Content Filter1 - renamed"
  }, {
    "filterUUID" : "5ba8cb68-b201-40f7-a1ab-13b947eaa933",
    "name" : "Test Content Filter1 - another edit"
  }, {
    "filterUUID" : "ca377171-9cde-4cd7-962e-5ab19632bc19",
    "name" : "Test Content Filter1 - edit3"
  } ],
  "count" : 9
}

```

POST Method Details

Add or update content filters in a specified config set.

Request

Name	Description
name	name of the filter.
rules	the array of rules defining a filter.

Response

Name	Description
errorCode	code for the change outcome, e.g. 'Success'.
errorDetails	details of an error, if applicable.
configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
serverGroupId	ID of the config set.

timestampSec	timestamp of the change.
modifiedBy	user who made the change.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/contentFilters -X POST -H "content-type: application/json" -d '{ "name" : "Test Content Filter", "rules" : [], }'</pre>
OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 94, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1677586225, "modifiedBy" : "admin" } }</pre>

Config Sets - Incident Filters

List the incident filters in a config set, and add or update incident filters.

URL: `cfg/serverGroupsConfig/{serverGroupId}/incidentFilters`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists the incident filters applied to a specified config set.
- POST Method - add or update incident filters in a specified config set.

GET Method Details

Lists the incident filters applied to a specified config set.

Response

Name	Description
filterUUID	unique filter identifier.
name	name of the filter.
selectedNames	array of incidents permitted by the filter.
passIP	if the filter should include devices not under management, either 'true' or 'false'.
count	number of filters.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/incidentFilters</code>
-------	---

```

OUTPUT {
  "items" : [ {
    "filterUUID" : "--AllEventsFilter--",
    "name" : "All Incidents",
    "passIP" : true
  }, {
    "filterUUID" : "4d76a68e-d960-438f-95b3-1cc395d2ac16",
    "name" : "Test Incident Filter",
    "selectedNames" : [ "ACI Event" ],
    "passIP" : false
  } ],
  "filterUUID" : "1a6cc6aa-f56a-4978-a16f-f239257373bd",
  "name" : "Test Incident Filter",
  "selectedNames" : [ "ACI Event" ],
  "passIP" : false
} ],
"count" : 3
}

```

POST Method Details

Add or update incident filters in a specified config set.

Request

Name	Description
name	name of the filter.
selectedNames	array of incidents permitted by the filter
flags	whether IP filters are passed.

Response

Name	Description
errorCode	code for the change outcome, e.g. 'Success'.
errorDetails	details of an error, if applicable.

configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
serverGroupId	ID of the config set.
timestampSec	timestamp of the change.
modifiedBy	user who made the change.

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/incidentFilters -X POST -H "content-type: application/json" -d '{ "name" : "Test Incident Filter", "selectedNames" : ["ACI Event"], "flags" : 1 }'</pre>
OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 96, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1677586487, "modifiedBy" : "admin" } }</pre>

Config Sets - Event Filters

List the event filters in a config set, and add or update event filters.

URL: `cfg/serverGroupsConfig/{serverGroupId}/eventFilters`

- [Methods Summary](#)
- [GET Method Details](#)
 - [Response](#)
 - [Example](#)
- [POST Method Details](#)
 - [Request](#)
 - [Response](#)
 - [Example](#)

Methods Summary

- GET Method - lists the event filters applied to a specified config set.
- POST Method - add or update event filters in a specified config set.

GET Method Details

Lists the event filters applied to a specified config set.

Response

Name	Description
filterUUID	unique filter identifier.
name	name of the filter.
selectedNames	array of events permitted by the filter.
passIP	if the filter should include devices not under management, either 'true' or 'false'.
count	number of filters.

Example

INPUT	<code>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/eventFilters</code>
-------	--

```

OUTPUT {
  "items" : [ {
    "filterUUID" : "--AllEventsFilter--",
    "name" : "All Events",
    "passIP" : true
  }, {
    "filterUUID" : "ce6dbc56-e662-4255-b7fc-14d8c2896582",
    "name" : "Test Event Filter",
    "selectedNames" : [ "ACI Event" ],
    "passIP" : false
  } ],
  "count" : 2
}

```

POST Method Details

Add or update event filters in a specified config set.

Request

Name	Description
name	name of the filter.
selectedNames	array of events permitted by the filter
flags	whether IP filters are passed.

Response

Name	Description
errorCode	code for the change outcome, e.g. 'Success'.
errorDetails	details of an error, if applicable.
configurationChangeId	ID of the current configuration change, referred to in the table under the <i>Sync History</i> tab on the Multi-Server Configuration page. Each time a config set is updated, the set gains a new configuration change ID. This ID is an auto-incrementing number.
serverGroupId	ID of the config set.
timestampSec	timestamp of the change.

modifiedBy	user who made the change.
-------------------	---------------------------

Example

INPUT	<pre>curl -u admin:admin https://localhost/api/cfg/serverGroupConfig/82e7c418-cd48-4fbf-857b-cddb475841a2/eventFilters -X POST -H "content-type: application/json" -d '{ "name" : "Test Event Filter", "selectedNames" : ["ACI Event"], "flags" : 1 }'</pre>
OUTPUT	<pre>{ "errorCode" : "SUCCESS", "errorDetails" : null, "data" : { "configurationChangeId" : 96, "serverGroupId" : "82e7c418-cd48-4fbf-857b-cddb475841a2", "timestampSec" : 1677586487, "modifiedBy" : "admin" } }</pre>